# HUMAN ASPECTS OF MACHINE LEARNING

A Dissertation
Presented to
The Academic Faculty

By

Samira Samadi

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science

Georgia Institute of Technology

May 2020

**HUMAN ASPECTS OF MACHINE LEARNING**

Committee members:

Dr. Santosh Vempala, Advisor
School of Computer Science
*Georgia Institute of Technology*

Dr. Adam Tauman Kalai
Senior Principal Researcher
*Microsoft Research New England*

Dr. Jamie Morgenstern
School of Computer Science & Engineering
*University of Washington*

Dr. Vivek Sarkar
School of Computer Science
*Georgia Institute of Technology*

Dr. Mohit Singh
School of Industrial & Systems Engineering
*Georgia Institute of Technology*

Date Approved: March 5, 2020

## ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Santosh Vempala, for giving me the opportunity to work with him, for his constant and reliable support over the past 5 years, and for patiently letting me explore a wide range of research. His guidance, passion for research, and the fruitful discussions that we had helped me to grow as a researcher. He taught me to think about problems fundamentally and to question my findings until I reach a vivid understanding of the problem. The skills that I developed through working with him have helped me throughout my Ph.D. and will continue to help me for the rest of my career.

I also want to thank my committee members, Adam Kalai, Jamie Morgenstern, Vivek Sarkar, and Mohit Singh, for serving as my thesis committee. Adam has been a great mentor, whose valuable insights helped me move forward with my research many times. Jamie helped me to get to know about fairness in machine learning, and her guidance and knowledge fast-forwarded my way into the field. Mohit is a brilliant and very helpful collaborator who is always generous with his time. Special thanks to Vivek Sarkar, his feedback on my work on password strategies gave me a better insight on how to improve my research and how to make it more usable.

I am grateful for my coauthors Pranjal Awasthi, Matthäus Kleindessner, Forough Poursabzi-Sangdeh, and Tao Tantipongpipat. Special thanks to Phil Long, Avrim Blum, Jenn Wortman Vaughan, and Hanna Wallach for hosting me as an intern. I am indebted to Manuel Blum for the time that he spent with me and for his ingenious mentorship.

I am thankful to many of my friends, without whom life in Atlanta would not have been the same. My deepest gratitude to my family for their love and everlasting support.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

## SUMMARY

With the widespread use of Machine Learning (ML) algorithms in everyday life, it is important to study the human aspects of these algorithms. ML algorithms are increasingly used in applications that influence our day-to-day life, both in the consumer and enterprise market. In the consumer market, machine learning is becoming the core part of most applications that we use in our life (e.g., ridesharing, social networks, etc.). Consumers expect to adopt these applications easily (*human-usable*) without sacrificing their privacy or security. In the enterprise market, ML-based applications are impacting consumers, for example, financial institutes use machine learning to assess the credit score of a loan application or evaluate a job application. Consumers demand visibility to how these decisions are made and expect a *fair* process.

All these ML processes and applications, directly or indirectly, interact with humans and thus it is crucial to study their effect. Such a study include, but is not limited to, studying and analyzing (i) the human usability of an ML system, and (ii) the influence of outcomes generated by such a system on humans. These studies can result in:

- ◇ Understanding of the limitations of users when adopting an ML system, and therefore improving its usability.
- ◇ Creating "fair" machine-generated outcomes, where fairness is defined relative to the context.

In this thesis, we study ML paradigms from the viewpoint of *human usability* and *fairness*. For human usability, we focus on two fundamental problems in password authentication:

(a) How can one generate humanly usable passwords that are secure?

(b) Given a limited memorization resource, what is the highest security that a humanly

usable password strategy could achieve? And can we construct such a password strategy?

To answer question (a), we introduced the first usability study of humanly computable *password strategies* (Chapter 2). To answer question (b), we showed that there exist humanly usable password strategies that are *hard* to hack; in fact, we showed that any adversary needs almost the information-theoretic number of samples to hack these strategies (Chapter 3).

For fairness, we studied representational bias in unsupervised learning settings such as the dimensionality reduction technique of principal component analysis (Chapter 4) and spectral clustering (Chapter 5).

# CHAPTER 1
# INTRODUCTION AND BACKGROUND

Machine Learning (ML) has recently seen an explosion of applications that directly or indirectly affect humans' decisions in everyday life. The demand for ML guided automated decision making has distinctly increased in major application domains including lending, marketing, education, and many more. Close on the heels of the adoption of ML methods in these everyday domains, there have been numerous concerns displaying the unsavory behavior of these systems towards humans. Such reports include but are not limited to compromises to users' information security, poor system interface design and deficient usability, and last but not least violation of ethical considerations towards various demographics. In this thesis, we take an step towards a rigorous analysis of a few ML paradigms from the viewpoints of information security, usability, and fairness.

For the first part of this thesis, we will focus on the fundamental problem of password authentication. We will closely study the security and usability of the very recently proposed password generation technique of *password strategies* [1]. For the second part of this thesis we will focus on fairness, and more specifically, the issue of representational bias in ML. We propose a new mathematical measure of algorithmic bias and study it in the context of dimensionality reduction. We conclude our work on representational bias by looking into a previously introduced notion of fairness [2], and showing how this fairness notion could be incorporated as a constraint into the spectral clustering framework.

## 1.1 Human Usability and Security of Password Strategies

Extensive research shows that many passwords in use can be easily guessed [3] and that people reuse passwords across different accounts [4, 5]. Password reuse leaves accounts vulnerable to a single breach. In order to generate secure passwords, users have to cre-

ate and remember complex strings, which often results in forgetting their passwords [6]. What makes this process even more tedious is that users are often forced to change their passwords. Unfortunately, the number of unique and secure passwords that users can comfortably memorize is very limited [7]. To overcome this limitation, most users tend to choose simpler passwords, or one strong password and use it across multiple websites. These approaches have resulted in many password breaches over the past few years [8, 9, 10, 11, 12].

**Password Strategies.** In an attempt to ameliorate these difficulties, recent work has introduced mental password strategies [13, 14, 1, 15] that enable users to systematically and securely generate and remember passwords for their different accounts. These strategies model passwords as mathematical functions from *challenges* (e.g., website names) to *responses* (character string passwords), and design such functions that can be computed by humans and are secure against a computationally powerful adversary. [1] defined *security* of a password strategy by: (i) given no prior information, how difficult it would be for an adversary to guess any generated password, and (ii) given that an Internet hacker has access to a few passwords that are generated using a specific password strategy, how difficult it would be to guess a new password generated with the same password strategy. They also proposed a precise model of human mental effort to measure the amount of *human computation* required in executing password strategies, but whether such methods are truly usable for most humans remained as an intriguing open question.

**Human Usability Study.** Password strategies may appear to be an appealing solution to the problem of remembering different passwords. Would human users (beyond mathematicians) find these methods pleasant? Would they be willing to adopt them? Several factors are important in the *usability* of a password strategy, including the amount of time that is required for learning and practice; memorizing the secret key; and using the strategy to generate a password. We designed a rigorous usability study to measure the effectiveness

of the following two password strategies

- ◇ *3-word strategy* which requires memorizing only three words and a special string. The letter map is then defined by mapping a letter $x$ to the consonant after (the first occurrence of) $x$ in the 3-word sequence. If $x$ is not present in the 3-word sequence, it gets mapped to a pre-assigned letter $z$.

- ◇ *random-letter strategy* which requires memorizing a random letter to consonant map and a special string.

For both of the above strategies, the password is the string generated by applying the map on the letters of the website name followed by the special string.

These password strategies are resilient to multiple breaches in the sense that even knowing multiple different challenge-password pairs, an adversary is unlikely to be able to guess one's password to a different challenge. Moreover, these strategies are *self-rehearsing* [14] in the sense that the process of typing passwords on different websites naturally reinforces the user's memory of the secret key.

In our empirical user study, we teach participants how to use a password strategy using videos (less than 5 minutes) that explain the concept and the problem being solved and teach them how to generate passwords using these methods. We then help them to choose their secret key and to memorize it and have them practice using the strategy on artificial website names. We later performed follow-up experiments simulating logins over the next month to evaluate how quickly and accurately participants can use their strategies on these and further artificial website names.

We find that for the random-letter strategy (3-word strategy), the teaching phase involved 5.2 (4) minutes of videos, a median of 8 (4.7) minutes to choose and memorize a secret key, a median of 6 (7.8) minutes to practice the strategy on 15 logins. On these and 24 other logins performed over the next month, the median time to enter a password was 2.9 (3.2) seconds per character, and the mean success rate of typing the correct password

within the first three tries was 98% (91%). Hence, there seems to be a trade off between learning and execution, with the 3-word strategy being faster to learn and memorize a secret key, while the random-letter strategy was faster to execute and gives higher accuracy.

### 1.2 An Optimal Security Password Strategy

Following the positive results of our initial human usability study on the 3-word and the letter-code strategy, we got motivated to explore other password strategies, especially ones with higher security guarantees. We ask the following fundamental question: Given a limited amount of memorization, what is the highest security guarantee that any password strategy can achieve? The next natural question is whether there exist humanly usable password strategies that achieve the highest possible information-theoretic security guarantee?

In Chapter 3 we investigate these questions in more details. We model the security of a password strategy as the sample complexity of learning the class of all such password strategies (with different secret keys). Given a limited amount of memorization, we bound the sample complexity of such a class of password strategies. Furthermore, we construct a humanly usable password strategy and show that it achieves the highest information-theoretic security guarantees. Our proposed password strategy, called the *digit* strategy, requires memorizing a random letter to digit map and a special string. The password for a website is then simply generated by summing the digit mappings of each letter of the website name $\pmod{10}$ and appending the special string.

We prove that any password strategy that requires an equivalent of $N$ digits of memorization, could achieve security guarantees of at most $N$. As we mentioned, for the digit strategy, the user needs to memorize a random letter to a digit map and a special string. This is roughly equivalent to $N = 26$ digits of memorization. We prove that assuming that website names are linearly independent the $26$-dimensional alphabet space, then the digit strategy achieves the security guarantee equal to $26$. We support our theoretical results by some empirical evaluations of the security of the digit strategy when tested on the top 400

website names.

## 1.3 Fair PCA

With the growing use of machine learning algorithms in automated decision making, researchers have raised concerns about the bias that these algorithms might produce in the outcomes [16, 17, 18, 19]. This has resulted in a wide range of studies focusing on detecting and eliminating sources of unfairness in different stages of a decision-making process, where most of this work has focused either on biased data or algorithms producing biased outcomes. In this regard, studying fairness for dimensionality reduction techniques focuses on a more subtle source of bias in ML applications, which may or may not be used in any particular decision-making process. When PCA is used as a preprocessing step for decision making, it can inadvertently erase critically useful information about some populations. Even when it is used merely to visualize data, the erasure of variance for some populations raises concerns of representational bias [20].

Principal Component Analysis (PCA) [21, 22, 23] is widely used as a preprocessing step to reduce the computational burden and/or to facilitate data summarization [24, 25]. We show on several real-world data sets, PCA has higher reconstruction error on one population than the other. This can happen even when the data set has a similar number of samples from each of these populations. This motivates our study of a dimensionality reduction technique which aims to represent multiple populations in the data with similar fidelity.

To achieve a fairer low dimensional representation of the data, we focus on finding a projection which minimizes the maximum *additional* reconstruction error for each population above the optimal $n$ into $d$ projection for that population alone. We refer to this quantity as *loss*.

**Definition 1.3.1** (Reconstruction loss)**.** *Given a matrix $Y \in \mathbb{R}^{a\times n}$, let $\widehat{Y} \in \mathbb{R}^{a\times n}$ be the*

*optimal rank-$d$ approximation of $Y$. For a matrix $Z \in \mathbb{R}^{a \times n}$ with rank $\leq d$, we define*

$$loss(Y, Z) := \|Y - Z\|_F^2 - \|Y - \widehat{Y}\|_F^2.$$

We ask to minimize the maximum average loss suffered by any group.

**Definition 1.3.2** (FAIR-PCA ). *Given $m$ data points in $\mathbb{R}^n$ with subgroups $A$ and $B$, we define the problem of finding a fair PCA projection into $d$-dimensions as optimizing*

$$\min_{U \in \mathbb{R}^{m \times n},\ rank(U) \leq d} max \left\{ \frac{1}{|A|} loss(A, U_A), \frac{1}{|B|} loss(B, U_B) \right\}, \tag{1.1}$$

*where $U_A$ and $U_B$ are matrices with rows corresponding to rows of $U$ for groups $A$ and $B$ respectively.*

This definition does not appear to have a closed-form solution (unlike vanilla PCA). To take a step in characterizing solutions to this optimization, we show that (Theorem 4.4.5) a fair PCA low dimensional approximation of the data results in the same average loss for both groups $A$ and $B$. Furthermore, we present a polynomial-time algorithm for solving the fair PCA problem (8). Our algorithm outputs a matrix of rank at most $d+1$ and guarantees that it achieves the fair PCA objective value equal to the optimal $d$-dimensional fair PCA value. The algorithm has two steps: first, relax fair PCA to a semidefinite optimization problem and solve the SDP; second, solve an LP designed to reduce the rank of said solution. We argue using properties of extreme point solutions that the solution must satisfy a number of constraints of the LP with equality, and argue directly that this implies the solution must lie in $d + 1$ or fewer dimensions.

The SDP and LP can be solved up to additive error of $\epsilon > 0$ in the objective value in $O(n^{6.5} \log(1/\epsilon))$ [26] and $O(n^{3.5} \log(1/\epsilon))$ [27] time, respectively. The running time of SDP dominates the algorithm both in theory and practice and is too slow for practical uses for the moderate size of $n$. We propose another algorithm for solving SDP using the

multiplicative weight (MW) update method. In theory, our MW takes $O(\frac{1}{\epsilon^2})$ iterations of solving standard PCA, giving a total of $O(\frac{n^3}{\epsilon^2})$ runtime, which may or may not be faster than $O(n^{6.5}\log(1/\epsilon))$ depending on $n, \epsilon$. In practice, however, we observe that after appropriately tuning one parameter in MW, the MW algorithm achieves accuracy $\epsilon < 10^{-5}$ within tens of iterations, and therefore is used to obtain experimental results. Our MW can handle data of dimension up to a thousand with running time in less than a minute.

**Generalization.** The definition of fair PCA could be generalized to the case that (a) there are more than two groups in the data and (b) for a broader class of social welfare functions. In Section 4.8 we will introduce the more general problem of multi-criteria dimensionality reduction and propose an efficient algorithmic solution for it. For the special case of fair PCA with two groups, our algorithm results in an exact solution, an improvement to the extra dimension that we discussed before.

### 1.4 Spectral clustering with fairness constraints

The paper of [28] has been the first to provide a notion of fairness for clustering. According to that notion, a clustering is fair if, in each cluster, every demographic group is represented with (approximately) the same fraction as in the whole data set. Think of a class of students in which half of the students are female and the other half are male and assume that we want to partition the students into several clusters to assign group projects. According to the proposed fairness notion of [28], such a clustering of the students would only be fair if each of the clusters consisted of about 50% female and 50% male students. The paper of [28], as well as some follow-up work [29, 30, 31], focuses on minimizing the $k$-center, $k$-median or $k$-means clustering objective while satisfying the fairness notion and hence requires the data set to lie in some metric / Euclidean space.

Many clustering problems involving human subjects, however, come in the form of a graph partitioning problem (e.g., when clustering the members of a social network), and for

these problems, spectral clustering (SC) is the method of choice in practice. In Chapter 5, we show how to incorporate the fairness notion of [28] into the SC framework. We utilize *constrained SC* and develop variants of both normalized and unnormalized constrained SC that aim to find a "good" (as measured by the SC objective) *and* fair clustering of a data set if such a clustering exists. On several real-world networks, we demonstrate that our algorithms indeed tend to find fairer clusterings compared to standard SC. Interestingly, while achieving a higher fairness level, the clusterings produced by our algorithms often only suffer from a minimal loss in the SC objective. An example of this finding can be seen in the left part of Figure 1.1.

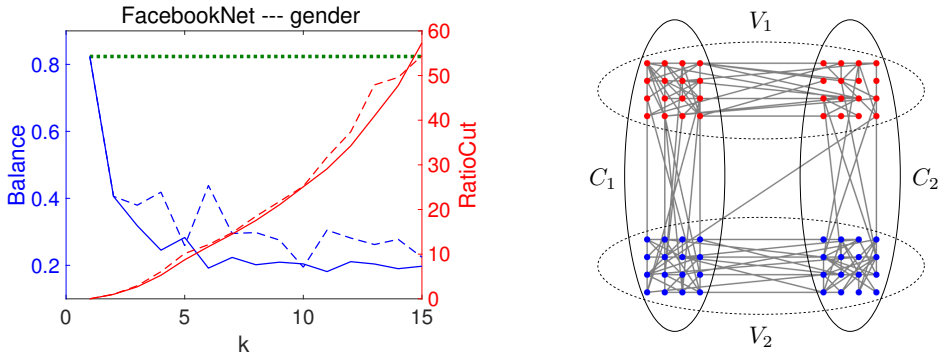



Figure 1.1: **Left:** Average balance of clusters and SC objective value (RatioCut) as a function of the number of clusters $k$ on a real-world social network. The full lines show the performance of standard SC, the dashed lines show the performance of our fair variant. **Right:** Example of a graph generated from our variant of the stochastic block model (with groups $V_1$ and $V_2$). There are two meaningful ground-truth clusterings of the vertices $V$ into two clusters: $V = C_1 \dot{\cup} C_2$ and $V = V_1 \dot{\cup} V_2$. Our algorithms return the first one, which is fair, while standard SC returns the second one, which is unfair.

We also provide a rigorous theoretical analysis of our algorithms on a natural variant of the stochastic block model, where the various demographic groups have strong inter-group connectivity, but also exhibit a "natural" clustering structure which is fair. An example of a graph generated from our model (with two groups and two clusters) can be seen in the right part of Figure 1.1. While for such a graph standard SC would simply return the two groups as clusters, we prove that our algorithms can recover the natural fair clustering with high probability.

# CHAPTER 2

# HUMAN USABILITY OF PASSWORD STRATEGIES

Reusing passwords across multiple websites is a common practice that compromises security. Recently, Blum and Vempala have proposed password strategies to help people calculate, in their heads, passwords for different sites without dependence on third-party tools or external devices. Thus far, the security and efficiency of these "mental algorithms" has been analyzed only theoretically. But are such methods usable? We present the first usability study of humanly computable password strategies, involving a learning phase (to learn a password strategy), then a rehearsal phase (to login to a few websites), and multiple follow-up tests. In our user study, with training, participants were able to calculate a deterministic eight-character password for an arbitrary new website in under 20 seconds.

The target audience of password strategies is, potentially, anyone who seeks a secure way to remember different passwords across many different accounts. The participants in our usability study were US-based crowd workers on Amazon's Mechanical Turk crowdsourcing platform, which has been shown to source a diverse set of users [32, 33] and often produce results similar to those of more traditional approaches [34]. Nonetheless, such users have a certain minimum age and demonstrated the ability to learn to perform tasks (we filtered for 98% task approval rating), which may differ from other groups of people using multiple accounts. More specifically, our participants reported being between 21 and 55 years old, with the gender distribution of 40% female and 60% male.

One experimental challenge is identifying whether (and how often) participants consult written or digital records of their secret keys. It is known that some people record passwords, and self-reporting cannot be relied upon, especially among Mechanical Turk workers [35] who have concerns about bonuses and future work. To reduce the utilization of written records, we provided participants with a button that would, with a single press,

remind them of their secret key while logging in. While this limits the ecological validity of our study, it enables us to track the frequency with which participants consulted this reminder. Participants varied in the frequency with which they pressed the secret key reminder button, though the frequency generally decreased (except among those participants that never pressed the button). The question of if users would keep records (and for what duration) is left for future work.

Password strategies may be a viable alternative to the common password management approaches of password reuse or writing passwords down. Another solution to the password memorization problem is to use a third-party password management software. Password vaults have become popular over the past few years as they require the user to remember only one master password and then the system automatically fills in login pages with strong (randomly generated) passwords. Unfortunately, this results in a single point of failure, which has caused security breaches [36]. Popular password vaults have been vulnerable to security attacks in recent years [37, 38]. Moreover, the user must install the vault on every device that she uses, making it difficult to use on shared devices such as a library computer or a friend's phone.

The rest of this chapter is organized as follows. In Section 2.1, we define the random-letter and 3-word password strategies. In Section 2.2, we describe the usability study in detail including the precise instructions given to the participants. Then we present the results of the user study in Section 2.3. In Section 2.4, we recall the human computation model of Blum and Vempala and use it to analyze the usability and security of the random-letter and 3-word password strategies. We discuss limitations of our study and mental password management strategies in Section 2.5 and present conclusions and future work in Section 2.6.

## 2.1 Mindhashes

Some of the password strategies require paper or digital assistance [13], but we focus on those strategies that can be computed in one's mind without any additional resources. For brevity, we use the term mindhash to refer to any such password management strategy that enables a user to mentally compute a different password for each challenge without external memory or computational aid, i.e., without paper or a smartphone [1]. Mindhashes require learning, memorization of a secret key, and execution when logging in to an account. In return for this effort, users enjoy security in the form of provable resilience to a small number of breaches. Blum and Vempala introduced several simple mindhashes with varying complexity, memory, and execution requirements, accompanied by varying security guarantees. We evaluate two of these mindhashes, one of which requires memorizing only three words. The strategies are resilient to multiple breaches in the sense that even knowing multiple different challenge-password pairs, an adversary is unlikely to be able to guess one's password to a different challenge. Moreover, these strategies are selfrehearsing [14] in the sense that the process of typing passwords on different websites naturally reinforces the user's memory of the secret key.

Here we describe two mindhash functions and approaches to choose and memorize their secret keys. We will use these mindhashes in our empirical and theoretical analysis. Both mindhashes consist of a map from letters to letters. To generate a password, this character map is applied to the challenge (website name) left-to-right, and a special character string is appended that meets various password-composition policies. For example, if the website name is six characters and the special string is three characters, then the password will be nine characters, consisting of the application of the character map to each of the six characters of the website name followed by the three-character special string. Blum and Vempala give more sophisticated mindhashes that have stronger security guarantees, but for the purposes of this study we restrict our attention to this character map type that still

offers significantly higher security than reusing a small number of passwords. Note that for actual use, small modifications are necessary for special cases such as non-alphabetical characters or very short domain names, as discussed in Section 2.5.

**3-word hash.** For this mindhash, the secret key consists of a user-selected 3 words that in total contain at least 15 different letters of the alphabet, a random letter (which we will refer to as a wild card), and a special character string consisting of an uppercase letter, a digit, and a non-alphanumeric character. The three words are concatenated to one single string, called the 3-word string. For example, one secret key is shown in Table 2.1. In this

| 3-word string | wild card | special string |
|---|---|---|
| *adjust flight computer* | *x* | *B7!* |

example, the 3-word string contains the 17 distinct letters $a\ c\ d\ e\ f\ g\ h\ i\ j\ l\ m\ o\ p\ r\ s\ t\ u$. The character map takes any letter *l* of the alphabet to the consonant that appears after the first occurrence of *l* in the 3-word string. In case that the letter *l* is not present in the 3-word string, then it maps it to the wild card. If the letter is the last consonant of the 3-word string, then it wraps around to the first consonant. Consonants are chosen because they offer greater entropy and hence greater security than vowels, which are more common and hence easier to guess.

Figure 2.1 shows an example of how to apply the 3-word mindhash. Suppose that you want to login to *amazon.com*. The challenge is the word *amazon*.

- Start with *a* (first letter of *amazon*) and find the first occurrence of *a* in *adjust flight computer*. Output the consonant that appears after *a*, which is *d* (Figure 2.1, box 1).
- The next letter is *m* and it appears in *computer*. The consonant after it is *p*. Output *p* (Figure 2.1, box 2).
- Repeat on the remaining letters of *amazon* (Figure 2.1, boxes 3-4).
- Append the special string *B7!* (Figure 2.1, box 5).

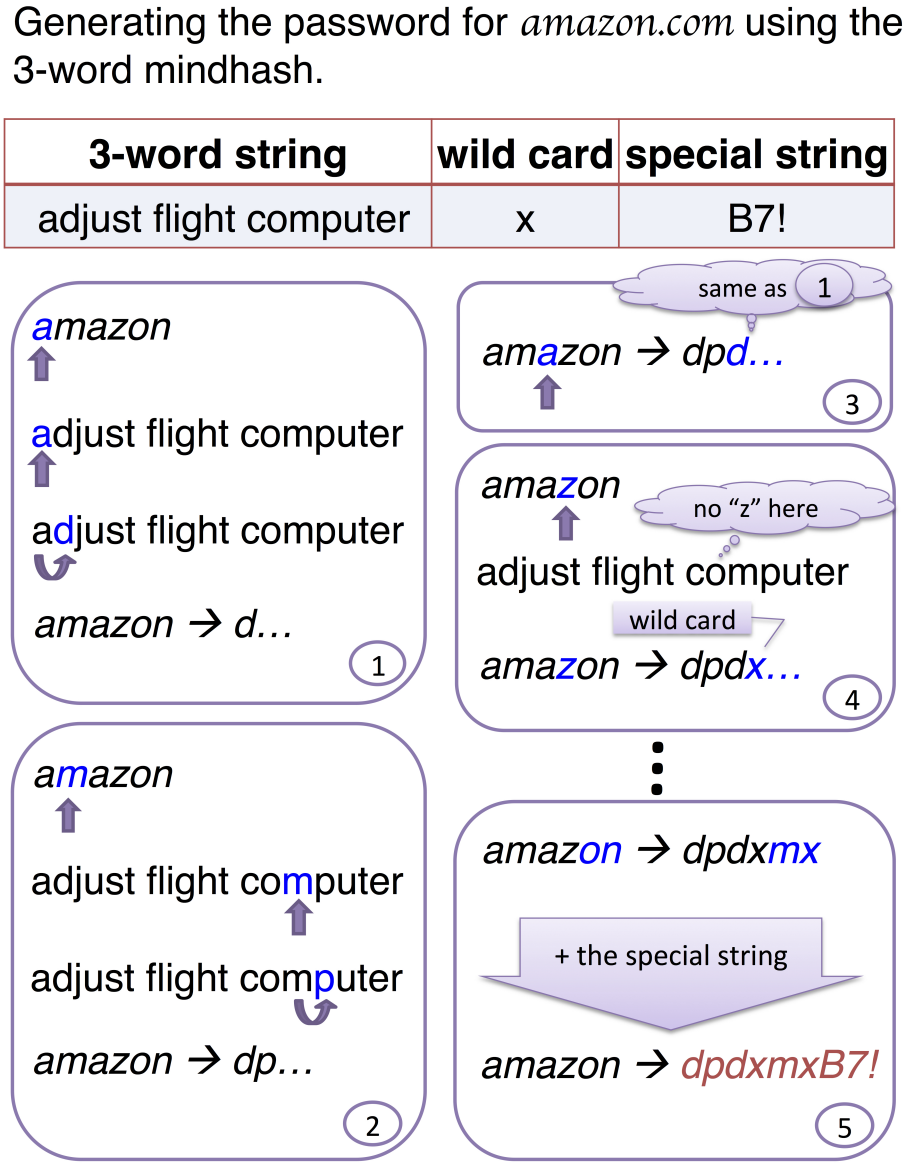



Figure 2.1: An example of generating a password using the 3-word mindhash. The top table shows the secret key and boxes 1-5 show the password generation step by step.

In Table 2.1, we show some examples of websites and their passwords.

Table 2.1: Passwords generated using our mindhashes.

| challenge | 3-word | Random-letter |
|---|---|---|
| *amazon* | dpdxmxB7! | qjqdr8*A |
| *facebook* | ldmrxmmxB7! | bqhgfddn8*A |
| *fidelity* | lgjrggfxB7! | blcgply8*A |

**Random-letter hash.** Like the 3-word hash, the random-letter hash is defined by a letter-to-consonant map and a special character string. Concretely, the user is aided in picking a **random** letter-to-consonant map for the first 20 letters of the alphabet (this truncation was done to increase usability – considering the fact that frequency of the letters *uvwxyz* is about $8\%$ in total) and choosing a special 3-character string that meets common password-composition policy requirements. If the challenge contains a letter from *uvwxyz*, the user skips that letter without any output. Alternatively, one could map each of these letters to a wild card, but since these letters are not common, this is not necessary and is not considered here. For example, consider the map

| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| q | f | h | c | g | b | s | k | l | m | n | p | j | r | d | t | n | w | x | y |

and the special string *8*A*. Suppose that you want to login to *amazon.com*. The challenge is the string *amazon*.

- Start with the first letter of the challenge *a* and find its mapping in the above table, *q*. Output *q*.
- The next letter is *m*, output *j*.
- Repeat on the remaining letters of *amazon*.
- Append the special string *8*A*.

See Table 2.1 for a few examples.

**Memorizing the 3-word hash.** The user memorizes the string of three words (order of words matters), a wild card and a special string.

**Memorizing the random-letter hash.** The user memorizes the letter hash using our method *Memorization with help of words*. The idea of this method is the following. The user looks at each letter pair, e.g., (a, q), and types the first word that comes to her mind

that starts with the first letter and has the target letter as the next consonant, e.g., aqua. She will do the same for all letter pairs (Table 2.2 left).

Table 2.2: Memorization with help of words.

| | | |
|---|---|---|
| a | q | aqua |
| b | f | beef |
| c | h | chef |
| d | c | duck |
| e | g | ... |

| | |
|---|---|
| a | aqua |
| b | beef |
| c | chef |
| d | duck |
| e | ... |

Note that the mnemonics do not, in fact, have to be English words, but can be any memorable strings. Once the words are written for all pairs, the user only needs to memorize the (first letter, word) associations (Table 2.2 right). This part should be rehearsed with repetition, i.e., rote memorization. Once the (letter, word) associations are memorized, the user can directly use them to recover the letter hash. Finally, the user memorizes the special string.

## 2.2 Human Usability Study Design

In this section, we describe the details of our usability study. Participants were randomly divided into two groups, with half of the participants being assigned to each mindhash. The reader can access and try all our surveys at the following link:
`https://github.com/PasswordUsability/Surveys`

**Qualification.** The participants had to pass a qualification test to be able to participate in our study. The qualification included reading a paragraph, informing about password security and then describing the study, followed by a few simple multiple choice questions. The qualification tested that participants were paying attention and understood the need for having different passwords for different websites.

**Learning the 3-word hash.** We showed the participants a short video[1] teaching them how to generate a password with a 3-word mindhash. After the video, to make sure they understood the idea, we asked them to generate passwords for one website using the same secret key that was used in the video tutorial. Participants were provided with the secret key, multiple attempts, and hints to aid in learning. At the end of this phase, participants learned how to generate passwords using a 3-word mindhash. After this phase, we asked the participants to choose their own three words sequence, wild card letter, and special string. Participants were allowed to proceed only if their word sequence contained at least 15 different letters and their special string contained an uppercase letter, a number, and a special character. As participants typed their 3 words, an alphabet letter bar, with the used letters crossed out, and the number of used letters was shown. This was to simplify the process of choosing words.

**Learning the random-letter hash.** We showed the participants a short video[2] teaching them how to generate a password using a random-letter hash. After the video, we displayed the letter map and the special string used in the video and asked them to generate two passwords. At this point, the participant did not need to memorize a letter map or a special string, but had to practice generating passwords using such a map. Next, we provided participants with an interface to choose a random letter-to-consonant map for the first 20 letters of the alphabet.

In the next step, we showed them a simple illustrative video[3] explaining our *memorization with help of words* technique. Then we ask the participants to repeat the letter pairs and the corresponding words for themselves. Although such a memorization might be done more quickly by speaking aloud, we asked the participants to type the letter pairs and words to ensure compliance. To further solidify memorization of the character map, we gave three further exercises:

[1] `https://youtu.be/HqFhimpK1iM`
[2] `https://youtu.be/DlPIy3WaAH4`
[3] `https://youtu.be/yd88MFZttvQ`

- Showing the letter pairs and asking the participants to type the words.
- Showing only the left letter and asking the participant to first type the word and then the right letter.
- The same as second exercise, but showing the left letters in a different order, e.g., "b, d, c, e, a" in Table 2.2.

**Practice.** Immediately after the learning phase, participants were presented with 15 artificial website names to try to log in, one at a time. For each website, they were asked to type the password using the mindhash that they had learned. Two hint buttons were provided. One showed text instructions on how to generate the password using the mindhash, and the other one displayed the participant's secret key.

Participants had three tries to type each password. If they failed in all tries, they were presented with the correct response.

**Feedback after learning.** Participants were asked to give us their feedback on different aspects of the study. We asked them if the task was fun/boring, easy/hard and if the password generation became easier toward the end, on a seven point bipolar rating scale. Finally, we asked the participants whether they would like to participate in our follow-ups.

**Follow-up evaluations.** After learning and practice (day 0), we performed six follow-up evaluations of the participants' ability to log in using their passwords, over a period of one month. The first follow-up was performed the next day (day 1), the second follow-up was again a day later (day 2), and the remaining four follow-ups were at day 4, day 8, day 16 and the final follow-up during days 32-35. The last follow-up was scheduled during a holiday period and thus we allowed the participants to fill it out anytime during a 3 day interval. At each follow-up, the participants were asked to generate passwords for 4 challenges. For each challenge, three attempts was given to type the password, and then

the correct password was shown.

Studies show that users manage on average 25 password-protected accounts [7]. Some of these accounts are used frequently (e.g., work account) and some are used occasionally. We consider 25 synthetic website names chosen as random common words: {kite, pillow, atlantic, bundle, reverse , family, quebec, cough, subject, mug, spike, fishing, jumper, knob, chord, quiz, fixed, world, campaign, warm, navy, banquet, hazy, chef, twist}. We assume that the first 15 names are frequent accounts and the last 10 are occasional or newly opened accounts. To reflect this, we asked the participant to type the passwords for all the frequent accounts at the end of the learning phase. The challenges in the follow-up evaluations were chosen with probability $75\%$ from the frequent accounts and with probability $25\%$ from the infrequent accounts, to reflect the use of passwords for both logging in to frequent accounts and infrequent or one-time accounts.

The 1/2/4/8/16/32-day timing follows a doubling schedule [39], which has been shown to be an effective repetition spacing in the practice of learning [40]. In addition to these sequential follow-ups, we ran a quantitative follow-up survey on day 4 of the study. In this survey, participants were asked to provide a self-recall of the secret key that they memorized.

**Hints and writing down passwords.** Since the study was performed online, one concern is that our results would be tainted by participants writing down their secret keys (or storing them in a file) and consulting this record, i.e., cheat sheet, without our knowledge in the experiment. Moreover, self-reporting cannot always be trusted in an environment like Mechanical Turk when users have motivations that keep them from being honest [35, 41]. In our case, workers do not know our experimental protocol and may hope for bonuses for "good" work, not to mention pervasive fears of unfair rejections. Moreover, workers who want to be on the good side of the requester for future work may try to impress the requester with their good memory. For all these reasons, it would be tempting for a worker

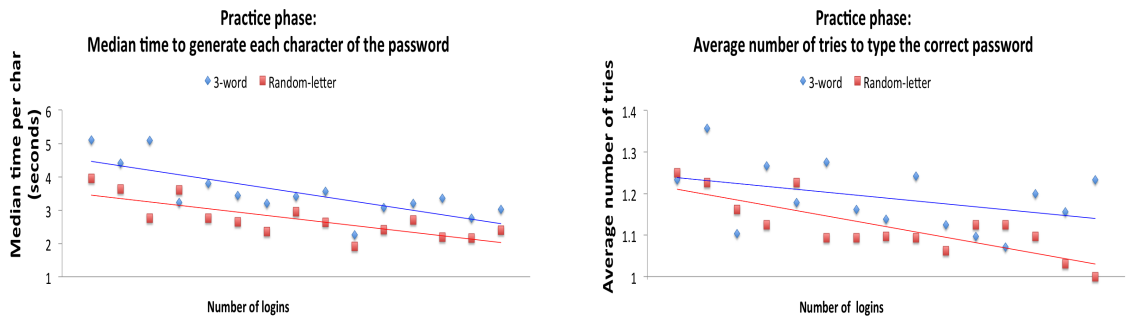



Figure 2.2: Practice phase, 15 logins. Left: Median time that the participants spent per character of the password. Right: Mean number of tries to type the correct password.

to record his secret key without admitting it to a requester, even if the requester claimed that there would be no consequence for reporting this. To address this, participants knew that throughout the study they had constant access to two hint buttons, one reminding them of the instructions and the other one reminding them of their secret key. Participants were told that there was no penalty or cost to use these hints, and they pressed the hint buttons liberally.

Although we understand that pressing a hint button on the screen is ecologically different than using a cheat sheet, we were hoping that capturing participants' usage of these hint buttons could give us some insight about the extent of users long-term dependence to the cheat sheets in real-life. The use of a written record may not constitute a serious security problem [42], and this argument is of course only stronger if the written record is only consulted during the first few days of learning the secret keys.

## 2.3 Results

In this section, we present the result of our user study. The participants were US-based crowd workers on Amazon's Mechanical Turk platform with at least a 98% task approval rating. Our participants reported being between 21 and 55 years old with average age of 31 years old. The gender distribution was 40% female and 60% male. For random-letter hash, overall 32 users participated in the training phase and 12 finished the last follow-up.

For the 3-word hash, overall 34 users participated in the training and 14 finished the last follow-up [4]. Look at Table 2.3 for more details.

Table 2.3: Number of participants in the original study (day 0) and follow-up surveys during the one month of study.

| mindhash/survey day | 0 | 1 | 2 | 4 | 8 | 16 | 32-35 |
|---|---|---|---|---|---|---|---|
| Random-letter | 32 | 27 | 27 | 24 | 14 | 14 | 12 |
| 3-word | 34 | 28 | 25 | 25 | 18 | 16 | 14 |

Table 2.4: For random-letter (3-word) mindhash, learning time includes watching a 2 (4) minute tutorial video, choosing a personal secret key, and practicing the mindhash on a few passwords. Memorization time includes watching a 3.5 (0) minute video describing the memorization technique, and using it to memorize the secret key. Password Generation time is calculated for typing a password of length 8.

| Time | Random-letter | 3-word |
|---|---|---|
| Learning+Memorization | 8+13 min | 11+0 min |
| Password Generation | 19 sec | 25 sec |

**Learning phase.** Learning times are reported in Table 2.4. The median times were 8 minutes to learn the random-letter hash and 11 minutes to learn the 3-word hash. The learning time for the 3-word hash was longer due to the longer training video (4-minute video versus 2-minute video). The memorization step for the 3-word hash was negligible. For the random-letter hash, the memorization time was 13 minutes, including a 3.5-minute video tutorial (see Section 2.2 for the details of memorization).

The median time that the participants spent on generating each character of the password decreases over time (Figure 2.2 left) with a mean of 2.3 seconds per character for the random-letter hash, and 3 seconds per character for the 3-word hash (computed over the last 5 logins). This corresponds to a password generation time of 19 seconds for the random-letter hash and of 25 seconds for the 3-word hash for a password of length 8 (Table 2.4).

[4]Participants were paid $2 to complete the qualification, $10 ($7) to complete the day 0 training for random-letter (3-word) mindhash, $2 for sequential follow-ups, and $4 for day 4 quantitative follow-up.

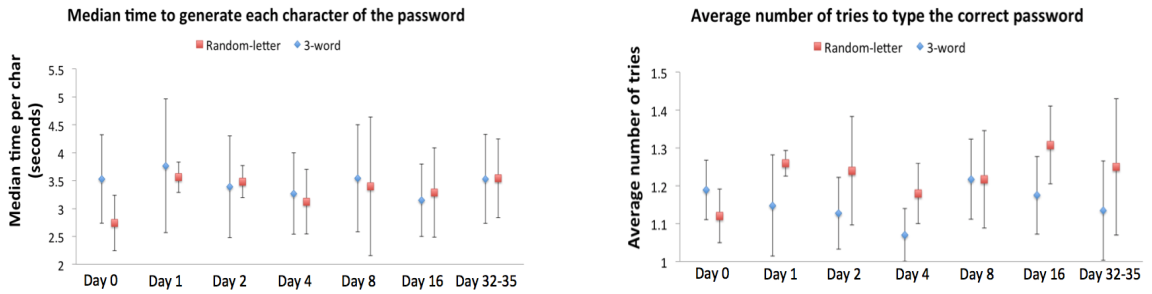



Figure 2.3: Left: Median time that participants spent generating each character of the password. Right: Mean number of tries for participants to login successfully. Error bars represent one standard error.

For both groups, the accuracy of typing the correct password during the practice phase was high: for each login, at least 96% (82%) of the participants typed correct passwords within three attempts using the random-letter (3-word) hash. Furthermore, the mean number of tries decreased over time (Figure 2.2 right).

In Figure 2.2 we see that, for both 3-word and random-letter mindhashes, the speed of generating each character of the password (left figure) and the accuracy of typing the password (right figure) increases as the number of logins increases. This improvement is an indicator that these mindhashes are self-rehearsing.

**Follow-ups.** Figure 2.3 left shows the median of the time that the participants spent on generating each character of the password each day. The error bars indicate the standard deviation of the medians, across participants, for all logins during the day. From prior work on memorization and self-rehearsing passwords, we hypothesized that the passwords generation time would decrease over time as the secret key and password process is establishing in long-term memory. Indeed, for both mindhashes, although the gaps between follow-ups doubled each time, password generation time remained low (less than 3.5 second/character during the last follow-up). This is an evidence that users could still type passwords reasonably quickly even for websites that are visited rarely. Figure 2.3 right shows the mean number of tries that participants needed to successfully login. Our result shows that, although the gaps between logins doubled over time, the accuracy of typing the

correct password for both mindhashes remained high over time (less than 1.3 attempts to successfully login). This is consistent with the self-rehearsable property of the password schemes.

As the participants type their passwords over time (follow-ups), we expect the secret key to be self-rehearsed and therefore participants to click on the secret key hint button less frequently. Figure 2.4 shows the fraction of participants that used the secret key reminder hint button.

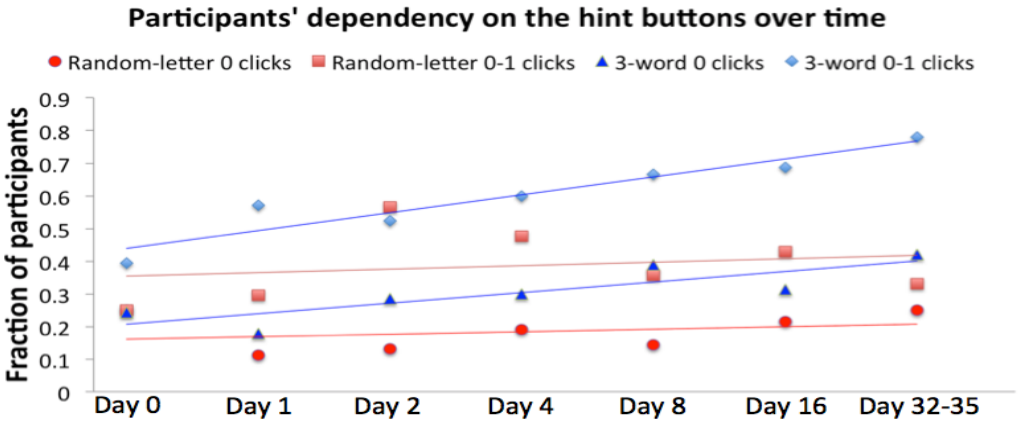



Figure 2.4: Fraction of participants that did not click on the hint button during logins and the fraction of participants that used the hint button for maximum one login.

Table 2.5 shows these results for the last follow-up. For 3-word hash 42% of the participants successfully typed passwords without the help of the hint buttons. For random-letter hash, the number is smaller, 25%, but still comprises a meaningful fraction of users. Note that participants were told that there was no penalty for using the hints, hence the actual use of such aids in practice would be expected to be lower. At the 4th day quantitative

Table 2.5: Fraction of participants that clicked on the hint button for the secret key during the last follow-up.

| No. click(s) | Random-letter hash | 3-word hash |
|---|---|---|
| 0 clicks | 25% | 42% |
| $\leq$ 1 click | 33% | 78% |

follow-up, participants were asked to type a free recall of their secret key. For the 3-word

hash, $70\%$ of the participants perfectly remembered the 3 words and $95\%$ of the participants remembered at least 2 words. For random-letter hash $31\%$ of the participants remembered at least 18 letters out of 20 ($90\%$ of what they memorized), and $78\%$ of the participants remembered at least 12 out of 20 letters ($60\%$ of what they memorized).

At the end of the follow-ups after a month, participants were asked if they had adopted the mindhash for generating passwords for managing their own personal passwords. For random-letter hash, 25% of the participants reported that they have adopted the mindhash in "real life". For the 3-word hash, 42% of the participants reported that they used the mindhash for generating their own personal passwords. Although such statistics are known to be greatly inflated, the comparison between the two schemes may be of interest.

**Feedback.** At the end of the training phase, we asked the participants to fill out the feedback form discussed in Section 2.2, and we further received free-text feedback throughout the one month study. For both mindhashes, participants reported that the effort for generating password decreased over time. Participants of both studies reported that they have found the task of generating passwords using mindhashes *neither easy nor hard*, with random- letter hash being slightly easier. Participants of random-letter hash found the task slightly fun. This was not completely the case for the 3-word hash as the participants reported that the task was neither boring nor fun. For both studies, 6% of the participants reported that they wrote down information.

Overall, participants found the random-letter hash study more fun and interesting. One reason was that they were surprised that they could memorize such a letter map: "This actually worked" or "Worked surprisingly well" were some of the comments that they provided. Over the one month period, participants got comfortable generating passwords and reported that the password generation is feeling more and more natural over time. Some typical anecdotal feedback that participants provided during the follow-up included:
**3-word hash:** "It's getting easier." or "It's definitely getting easier. I still have to open my

word list but my brain is adapting and I'm starting to know what each letter should translate to without looking sometimes.".

**Random-letter hash:** "I have definitely warmed up to the program. It feels more natural now than the last time. " or "I think I've finally got a handle on this password combination! Well, minus the one mistake.".

### 2.4 Theoretical analysis

Usability of a mindhash has two main aspects: learning time and password generation time. In this section, we discuss the rigorous model from Blum and Vempala for the password generation time.

**Password generation time** is the time that the user spends on outputting her passwords. Password generation is done entirely in the human's head with no paper, writing instrument, or computing device. It can be viewed as a restricted streaming computation. The working memory [43] is very small, typically at most one or two pointers and two characters (which might typically be letters or digits). Each elementary operation (retrieve a sequence from long-term memory, follow a pointer, add two digits mod 10) has a cost, which is the total number of write operations to the working memory. For example, retrieving a pointer to a sequence in long-term memory has cost $1$, following the sequence has cost $1$, adding two digits $\pmod{10}$ has cost $1$ or $2$ depending on the number of digits created. A human algorithm can thus be assigned a total cost, by adding up the cost of each step. This is the human complexity of the algorithm (called Human Usability Measure or HUM in Blum and Vempala). It is meant as a complexity measure for human computation analogous to the standard runtime complexity analysis of Turing machines. The HUM measures the human effort required to execute algorithms. Just as machines running the same algorithm can take different times, humans also have variability in speed.

**Password Generation Phase.** Given a challenge $c = c_1 \dots c_n$, start with the first letter $c_1$. Output the mapping of $c_1$. If mapping of $c_1$ is not defined, don't output anything or output

the wild card (depending on the mindhash's instruction). Shift the pointer to the next letter and do similarly for the remaining letters. Append the special string $s$ to the end of your password. To illustrate this measure, we now compute the HUM for the 3-word and the random-letter hash.

**HUM of 3-word hash.** Let $W_1W_2W_3$ be the sequence of words and $s$ the special string. Let $f$ be the letter-to-letter map defined by the 3-word hash. The cost of applying $f$ is initially higher (to scan the words and find the next consonant) but finally becomes 1.

**Algorithm 1:** 3-word hash

**Input**: Challenge $c = c_1 \dots c_n$
Retrieve challenge $c$. Pointer $\rightarrow c_1$. Cost = 1
**while** *not end of* $c$ **do**
  Let $c^*$ be the current character.
  Output $f(c^*)$ Cost = 1
  Shift pointer to next character. Cost = 1
**end**
Retrieve fixed string $s$. Pointer $\rightarrow s_1$. Cost = 1
**while** *not end of* $s$ **do**
  Output current character. Cost = 1
  Shift pointer to next character. Cost = 1
**end**

The HUM is $1 + n(1 + 1) + 2|s| = 2n + 2|s| + 1$.

**HUM of random-letter hash.** Similar to Algorithm 1, we can write the algorithm that describes the HUM of the random letter hash. The HUM is $2n + 2|s| + 1$.

**Security.** Password strategies should be secure against a computationally all-powerful adversary observing (challenge, response) pairs and trying to impersonate the human. We use the following two security parameters, identified in earlier work [1].

It should be hard for the adversary to guess any password of the user. This is the intuition behind the definition of the security parameter $K$. Given a password strategy $\mathcal{S}$ and a positive integer $i$, we say that $K_{\mathcal{S}} = i$ if for any single challenge $c$, the probability that an adversary can guess the correct response to $c$ is at most $1/10^i$.

**Algorithm 2:** Random-letter hash

**Input:** Challenge $c = c_1 \dots c_n$

| | |
|---|---|
| Retrieve challenge $c$. Pointer $\rightarrow c_1$. | Cost = 1 |
| **while** *not end of* $c$ **do** | |
| Let $c^*$ be the current character | |
| Output $f(c^*)$ | Cost = 1 |
| Shift pointer to next character | Cost = 1 |
| **end** | |
| Retrieve fixed string $s$. Pointer $\rightarrow s_1$. | Cost = 1 |
| **while** *not end of* $s$ **do** | |
| Output current character | Cost = 1 |
| Shift pointer to next character | Cost = 1 |
| **end** | |

Assume that an internet hacker has found your password to a couple of insecure websites, and is trying to login to your bank account. She might not have the full information to precisely guess your bank account password, but she will have partial information that narrows her predictions to $4$ choices. As a result, if your bank website allows her to try multiple guesses, she can successfully login to your account. How many tries will she need? How many passwords will she need to see in the clear? This is the motivation behind the definition of the security parameter $Q$. Given a password strategy $\mathcal{S}$ and $P \in (0, 1)$, $Q_{\mathcal{S}}(P)$ is defined as the number of random (challenge, response) pairs that an adversary must observe in order to be able to respond correctly to the next challenge with probability greater than $P$. The dictionary of challenges must be specified (e.g., English words, random strings, the top 500 most popular website names, etc.).

$K_{\textbf{random-letter}}$. Given a challenge $c = c_1 \dots c_n$, the adversary can respond correctly to $c$ only if she can correctly guess the mappings for all the letters of $c$ and the special string $s$. Each random letter has been chosen uniformly at random from the set of 21 consonants. The user's special string consists of one capital letter, one number and one special character, all chosen uniformly at random too[5]. Therefore, the probability that the adversary can guess

[5]This assumption is based on the distribution of special strings reported by the users.

the correct password is

$$Pr[\text{guess}(c) = \text{password}(c)] \leq (1/21)^n(1/26)(1/10)^2.$$

Assuming that an average password has length 8 (challenge of length five characters[6]), this gives us $K_{\text{random-letter}} \geq 10$.

$Q_{\textbf{random-letter}}(P)$**.** The adversary can respond correctly to a challenge only if she has seen all letters in the challenge in the previous challenges. If she has not seen even one letter, the chance of guessing the correct response to the challenge is $1/21$. What are the expected number of (challenge, response) pairs that the adversary should see to have complete knowledge of the mapping of all letters of a random new challenge? For the top 500 domain names, this value is equal to $6.6$ [1]. Therefore, for any $P \geq 1/21$, $Q_{\text{random-letter}}(P) \geq 6.6$.

Most secure websites block the user's account if he types a wrong password for 3-5 times. This is equivalent to $20\% < P < 33\%$, and thus the above security parameter value is meaningful. Also note that once the adversary sees one password, she already knows the special string $s$. Therefore $s$ does not contribute to $Q$. The security of the 3-word hash is lower since the total entropy generated by choosing 3 random words is smaller. $Q_{\text{3-word}}$ is estimated as between $3$ and $4$ [1].

### 2.5 Limitations

We have shown that mindhashes are secure and human-usable solutions for choosing passwords for many users. However, in this section, we discuss limitations of mindhashes and the study that we have done in this paper.

**Password policies.** Websites have policies with different password requirements involving password length or special characters. In our study, users were instructed to append a fixed "special string" to all of their passwords in order to meet such requirements. A recent

[6]For longer challenges, the user can use only the first 5 characters of the challenge.

survey finds that it is often possible to choose a single such string that simultaneously satisfies the requirements of different websites [44]. However, in special cases, a website may have different requirements that may not be met by the special string. In general, this is considered as a challenging problem for other password generating approaches as well [45].

**Short or irregular challenges.** Some website names may be very short or contain non-alphabetic characters, such as 53.com for the *Fifth Third Bank*. While not measured in our study, it would be natural for users to choose a memorable, sufficiently long challenge for these websites, such as the string *fifththird*. Note that different users may choose different challenges, but this does not cause any problems as long as each user is consistent with using the same challenge. Further study is necessary to see how common this problem is and how easy it is for users to recall their challenges.

**Infrequently used accounts.** Our study does evaluate the ability to correctly generate passwords for numerous new challenges, which is similar to generating a password for a rarely visited website. The 3-word hash has a natural self-rehearsing property so that using it frequently reinforces the memory of the entire secret key, and hence generating passwords for rarely used challenges is straightforward. However, for the random-letter hash, *infrequently used letters* pose a greater problem. For example, a user may forget her mapping for the letter $q$ if it is never used.

**Passwords sharing.** Sharing passwords across different accounts is a problem that is not addressed by the mindhashes. Although mindhashes do not offer any solution for sharing passwords across different accounts, if a user chooses to share a password, security is not entirely compromised.

**Changing passwords.** Certain systems may require passwords to be changed periodically. This is a problem with the password management that is not studied in our work and is not directly addressed by mindhashes. A solution for this, suggested in Blum and Vempala, is to append a digit that indicates which letter of a challenge the user should start with when

generating a password. The human usability of these approaches can be studied as part of future work.

**Entropy of passwords.** Mindhashes assume that the secret key is chosen randomly. For example, in the random-letter strategy, we assume that the user memorizes a random letter-to-letter map. Although we provide a simple interface for users to build such a random map, it is still possible that in practice users may choose predictable secret keys (e.g., for the letter *a* some letters may be more commonly chosen, such as *p* for *apple*, or a person named *Alice* may be more likely to choose *l*). This would reduce the entropy and advantage an adversary that attempts to guess the secret key.

**Multiple accounts on the same website.** Some users may have multiple accounts on one website. In this case, they may use the same password across accounts.

**Dropouts and hints.** For both mindhashes, approximately 60% of participants dropped out during the course of the study. Our statistics should be interpreted as representative of the 40% of participants who completed the study. While we could have provided additional incentives in the form of completion/milestone bonuses to increase completion rates, we felt that there was value in observing the natural completion rate at a static pay rate. As discussed, participants had the opportunity to press a hint button to see their secret keys without any discouragement or adverse affect on their payment. In the last follow-up, 25% (42%) of the participants using the random-letter (3-word) mindhash did not use hints even once. Taken together, if one considers mindhashes "usable" for such participants, this gives a **lower bound** of 9% (18%) on the usability rate. This is a lower bound because it is likely that some users did not complete the study for various personal reasons aside from usability, and that some users clicked the hint buttons even when they would have found the system usable without hints. We provided the hint button to dissuade users from secretly recording their secret keys in a way that we could not monitor.

## 2.6 Conclusion

We presented the first user study of two different mindhashes (i.e., password strategies): *3-word hash* and *random letter hash*. Participants in our user study spent a median of 11 minutes learning the 3-word hash and 8+13=21 minutes learning the random-letter hash. After the learning phase, the user is ready to use these mindhashes, and it takes 19-25 seconds to generate a password. As predicted by the self-rehearsing property of mindhashes, the time to generate a password decreases over time. We showed that, although there are increasing gaps between rehearsals with no practice, users remembered their secret key and were able to successfully login to arbitrary websites. Although the presence of the reminder button decreases ecological validity, users consulted the button with decreasing frequency. It was encouraging that some users seemed interested in adopting these methods to manage their own passwords. A natural research question is to identify mindhashes with even better usability and security.

**Open Questions and Future Work.** I adopted the random-letter strategy to enhance the security and usability of my passwords three years ago. Since password strategies had a big impact on my personal life, I have been promoting them through `www.safepasswords.org`. There are yet major questions left to be studied for large-scale adoption of these methods. Although we provide platforms through which the users can generate random secret keys, many users might not use these platforms. How much do self-created secret keys reduce our security guarantees? Furthermore, our proposed password strategies might not be usable for all the users, e.g., some users might remember visual images better than words. Can we design a platform through which the users could not only learn, but also be able to design their password strategies?

# CHAPTER 3

# HUMANLY USABLE PASSWORD STRATEGIES THAT ARE OPTIMALLY SECURE

Passwords are essential and ubiquitous. Password methods currently in use are either alarmingly insecure or humanly unusable, needing an inordinate amount of effort on the user's part [46, 47, 48]. In [1], it was shown that there exist password strategies that are (a) precisely defined, i.e., there is no ambiguity about how to create or re-generate passwords (b) humanly usable in a rigorous model of human computation and (c) secure to a well-defined extent, where the security is against a computationally all-powerful adversary, assuming the adversary (she) knows the class of password strategies that the human (he) is choosing his strategy from, but not his specific strategy. An important question raised is the following:

*What is the sample complexity of learning a class of humanly usable password strategies? How many passwords does an adversary need to see to impersonate the user?*

A password mindhash, as we introduced in the previous chapter, should be computable by a human in his/her head without using paper/pencil/computer, i.e., the algorithm to compute the function has to work in a highly restricted model of computation. They should be secure against a computationally all-powerful adversary observing (challenge, response) pairs and trying to impersonate the human. Security of a password mindhash can be captured by the number of randomly chosen (challenge, response) pairs that must be observed in order to exceed a desired threshold probability of responding correctly to the next challenge. This measure of security is closely related to sample complexity of weak learning the user's mindhash.

We show that, rather surprisingly, there exist humanly usable password mindhashes that are hard to learn; in fact, any adversary needs almost the information-theoretic number

of samples to learn these functions, and the strategies are optimally secure for any given amount of private randomness (memorization). Before stating our results, we must make precise what we mean by *humanly usable* and define the theoretical setting for measuring sample complexity.

### 3.1 Human Computation Model

A model for human computability was defined in [1]. There are two aspects: (1) Preprocessing and (2) Processing.

*(1) Preprocessing* or memorization concerns the human's long-term memory, and takes time and effort [49] roughly proportional to what must be memorized. A password mind-hash must specify what to memorize and how to do so. Paper and pencil are allowed in the preprocessing phase.

*(2) Processing* or runtime computation is done entirely in the human's head with no paper, pencil or computing device. It can be viewed as a restricted streaming computation. The working memory [43] is very small, typically at most one or two pointers and two characters (which might typically be letters or digits). Each elementary operation (retrieve a sequence from long-term memory, follow a pointer, add two digits mod 10) has a cost, which is the total number of write operations to working memory. E.g., retrieve pointer to a sequence in long-term memory has cost $1$, follow the sequence has cost $1$, add two digits $(\mathrm{mod}\ 10)$ has cost $1$ or $2$ depending on the number of digits created. A human algorithm can thus be assigned a total cost, by adding up the cost of each step. This is the human complexity of the algorithm (called Human Usability Measure (HUM) in [1]).

We give one illustrative example. Suppose the input is a name ("Peter Parker") and the output is the parity of the number of letters in the name ($= 1$). The human algorithm would be

It costs $1$ to retrieve a pointer to the name (i.e., to the first letter of the name, e.g., "P" in "Peter Parker"), then cost $1$ to add, cost $1$ to shift pointer to the next letter, and $1$ to

**Algorithm 3:** Parity algorithm

**Input**: Challenge $c$ : "Peter Parker"
Set pointer to first letter of $c$
Set COUNT$= 0$
**while** *not end of* $c$ **do**
  Add $1$ to COUNT $\pmod 2$.
  Shift pointer to next letter.
**end**
Output COUNT.

output. The last letter of the name can be assumed to point to end-of-Name. Thus, the HUM $= 1 + n(1 + 1) + 1 = 2n + 2$.

The HUM measures the human effort required to execute human algorithms. Just as machines running the same algorithm can take different times, humans also have variability in speed. For human computation, asymptotic $O(.)$ complexity is too coarse, and the leading constants are important.

### 3.2 Sample Complexity of weak learning

Password mindhashes are functions that map *challenges* (strings of characters such as website names) to *responses* (strings of characters). The problem of learning a class of password mindhashes is closely related to the weak-learning protocol [50, 51, 52]. In this section, we will exploit this connection in more details.

Assume that an internet hacker has found your password to a couple of insecure websites, and is trying login to your bank account. She might not have full information to precisely guess your bank account password, but she will have partial information which narrows her predictions to $4$ choices. As the result, if your bank website allows her to try multiple guesses, she can successfully login to your account. How many tries will she need? How many passwords will she need to see in the clear? This is the motivation behind the definition of the "Password Game" and the analysis of weak PAC learnability [50].

We are given a set of challenges $\mathcal{C}$, set of responses $\mathcal{R} = \{r_1, \ldots, r_k\}$ and a finite

class of password mindhashes $\mathcal{H} = \{h : \mathcal{C} \to \mathcal{R}\}$. Let $D$ be a probability distribution on $\mathcal{C}$, $r \in \mathcal{H}$ be the target function and $0 \leq p \leq 1$ be the threshold probability of success i.e., the learner (adversary) succeeds if she correctly classifies a random challenge with probability higher than $p$. The password game and weak PAC learning is precisely defined as following:

**Algorithm 4:** Password Game

**Input:** $p \in [0,1], D, \mathcal{H}$
**Initialization**: $t = 1, p_t = 0$
**while** *true* **do**
  The learner gets a random challenge $c_t \sim D$
  The learner chooses a hypothesis function $h_t \in \mathcal{H}$
  Let $p_t = \Pr_{c \sim D}[h_t(c) = r(c)]$
  **if** $p_t > p$ *:* **then**
    game ends
    The learner is told the correct response $r(c_t)$
  **end**
  $t = t + 1$
**end**

**Definition 3.2.1** (Weak PAC learning)**.** *Given $p \in [0,1]$ and a class of password mindhashes $\mathcal{H}$, we say that $\mathcal{H}$ is $(p,\delta)$-**PAC learnable** with $m_{\mathcal{H}}(p,\delta)$ samples if there exist a learning algorithm $L$ and a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ such that for any $\delta \in (0,1)$ with probability higher than $1-\delta$ the learner $L$ wins the password game (weak learns with probability at least $p$) in $m \leq m_{\mathcal{H}}(p,\delta)$ steps (examples).*

The purpose of this study is two-fold. First, we aim to provide a rigorous mathematical model for studying password mindhashes, both their usability and security. Our second goal is to use this measure to design password mindhashes that can effectively be used in practice while having provable guarantees.

We begin with a simple upper bound on the sample complexity of weak learning any finite class of password mindhashes, assuming that the learner has unlimited computation power. Theorem 3.3.1 proves that any finite class of password mindhashes $\mathcal{H}$ is weak $(p,\delta)$-PAC learnable with less than $\lceil \frac{\log|\mathcal{H}|+\log 1/2\delta}{\log 1/p} \rceil$ samples. Initially, it might seem intriguing to

have password mindhashes with such high security, but it is important to note that this result is information theoretic and it does not provide any guarantee on the human usability of such mindhashes. Thus, the important question to study is the following:

**Question.** What is the sample complexity of weak learning a class of humanly usable password mindhashes?

The main result of this chapter is a constructive proof of a class of humanly usable password mindhashes $\mathcal{G}$ with near-optimal security against a computationally all-powerful (unbounded) adversary. Moreover the algorithm used is public. Mindhashes generate passwords of length $K$, for any desired integer $K \geq 1$. Roughly speaking, the guarantee says that for $p = 1/10$, $\mathcal{G}$ is not weak $(p, \delta)$-PAC learnable with less than $N - \log(\frac{1}{(1-\delta)})$ samples, where the number of mindhashes is $10^N$. Here $N$ can be viewed as the number of digits of memorization needed. And, as we show, the mindhash can be learned efficiently using $N + \log(2/\delta)$ examples (challenge-response pairs). Thus, the lower bound for a humanly-usable class of mindhashes nearly matches the information-theoretic bound up to an $\log(\frac{1}{\delta(1-\delta)})$ additive factor, for any desired success probability $\delta$. We supplement these theoretical results with experiments demonstrating that the security guarantees are indeed realized even for very small parameter values.

### 3.3 Upper bound

The main result of this section is Theorem 3.3.1, providing an upper bound on the sample complexity of weak learning any finite class of password mindhashes.

**Theorem 3.3.1.** *Given a class of finite password mindhashes $\mathcal{H}$ and probability threshold $p \in (0, 1)$, $\mathcal{H}$ is weak $(p, \delta)$-PAC learnable with $\lceil \frac{\log |\mathcal{H}| + \log 1/2\delta}{\log 1/p} \rceil$ samples.*

In order to prove this, we need to define the notion of Random-Vote and Random-Voter.

**Definition 3.3.2** (Random Vote)**.** *Given $\mathcal{G} \subseteq \mathcal{H}$, A Random-Vote on $\mathcal{G}$ is defined as following. For every challenge $c \in \mathcal{C}$, let $\mathcal{G}_i = \{h \in \mathcal{G} \mid h(c) = r_i\}$ for $i \in [k]$*

$$\textit{Random-Vote}_{\mathcal{G}}(c) := \begin{cases} r_1 & \textit{w.p} \quad \frac{|\mathcal{G}_1|}{|\mathcal{G}|} \\ \vdots & \\ r_k & \textit{w.p} \quad \frac{|\mathcal{G}_k|}{|\mathcal{G}|} \end{cases}$$

**Algorithm 5:** Random-Voter

**Initialization:** $t = 1, \mathcal{H}^1 := \mathcal{H}$
**while** *True* **do**
  The leaner gets a random challenge $c_t \sim D$
  The learner chooses $h_t = \text{Random-Vote}_{\mathcal{H}^t}$
  Let $p_t = \Pr_{c \sim D}[h_t(c) = r(c)]$
  **if** $p_t > p$ **then**
    game ends
    The learner is told the correct response $r(c_t)$
  **end**
  $\mathcal{H}^{t+1} = \{h \in \mathcal{H}^t | h(c_t) = r(c_t)\}$
  $t = t + 1$
**end**

*Proof.* (Theorem 3.3.1) Assume that the password game has ended after $k$ steps. It means that in the steps $1, \ldots, k$ the probability that Random-Voter predicts the correct response for a random challenge is less than $p$.

$$\mathbb{E}_{c_1 \sim D}[|\mathcal{H}^1|] = |\mathcal{H}| \mathbb{E}_{c_1 \sim D}[|\frac{\mathcal{H}^1}{|\mathcal{H}|}|] < p|\mathcal{H}|$$

So after the first round of the game, in expectation the number of functions that survive

to the next round is less than $p$ fraction of all the functions

$$\begin{aligned}
\mathbb{E}[\mathcal{H}^2] &= \mathbb{E}_{c_1\sim D}[\mathbb{E}_{c_2\sim D}[|\mathcal{H}^2||\mathcal{H}^1]] \\
&= \mathbb{E}_{c_1\sim D}[|\mathcal{H}^1|\,\mathbb{E}_{c_2\sim D}[\frac{|\mathcal{H}^2|}{|\mathcal{H}^1|}|\mathcal{H}^1]] \\
&= \mathbb{E}_{c_1\sim D}[|\mathcal{H}^1|\,p] \\
&= p\mathbb{E}_{c_1\sim D}[|\mathcal{H}^1|] \\
&\leq p^2|\mathcal{H}|
\end{aligned}$$

Similarly, one can see that after $k$ rounds of the game $\mathbb{E}[\mathcal{H}^k] < p^k|\mathcal{H}|$. Using Markov inequality we get that

$$\Pr[|\mathcal{H}^k| \geq 2] \leq \frac{\mathbb{E}(|\mathcal{H}^k|)}{2} \leq \frac{p^k|\mathcal{H}|}{2}$$

Note that the game will definitely end if only one function survives. So we want the left-hand side to happen with low probability. Setting the right-hand side to be smaller than $\delta \in [0,1]$, it is easy to see that for $k \geq \lceil \frac{\log|\mathcal{H}|+\log 1/2\delta}{\log 1/p} \rceil, \Pr[|\mathcal{H}^k| \geq 2] \leq \delta$. Equivalently, with probability higher than $1-\delta$, after $\lceil \frac{\log|\mathcal{H}|+\log 1/2\delta}{\log 1/p} \rceil$ rounds of the game, only one function survives. □

### 3.4 Lower Bound

So far we have shown that random voter weak $(p,\delta)$-PAC learns a class $\mathcal{H}$ of password mindhashes with $\lceil \frac{\log|\mathcal{H}|+\log 1/2\delta}{\log 1/p} \rceil$ samples. In this section, we give a constructive proof of a class of password mindhashes that are humanly usable and any adversary with unbounded computation power, can't weak learn it with fewer samples than Random-Voter.

To state our main result we need some definitions. We view a challenge as a vector with integer coordinates in $\Re^N$, one coordinate for each letter of the alphabet, where the coordinates are ordered according to some fixed ordering of the letters of the alphabet.

For a prime $p$, a set of challenges is linearly independent $(\mathrm{mod}\ p)$ if their corresponding vectors are linearly independent $(\mathrm{mod}\ p)$. For an arbitrary positive integer $a$, with prime factors $p_1, p_2, \ldots, p_r$, we say that a set of challenges is *linearly independent* $(\mathrm{mod}\ a)$ if the vectors corresponding to the challenges are linearly independent $(\mathrm{mod}\ p_1)$, $(\mathrm{mod}\ p_2)$, ..., and $(\mathrm{mod}\ p_r)$. In particular this captures linear independence $(\mathrm{mod}\ 10)$ as being linear independent $(\mathrm{mod}\ 2)$ and $(\mathrm{mod}\ 5)$.

**Theorem 3.4.1.** *There exists a class of humanly usable password mindhashes $\mathcal{G}$ that*

1. *The HUM complexity for a challenge of length $n$ is at most $2K + 3.5n$ where $K$ is the length of responses.*
2. *For $p = 1/10$ any learning algorithm needs at least $\lceil \frac{\log|\mathcal{G}| - (K-1) - \log(\frac{2}{1-\delta})}{log 1/p} \rceil$ samples to weak $(p, \delta)$-PAC learns $\mathcal{G}$.*

The first part of Theorem 3.4.1 is proved in Section 3.5. The second part of Theorem 3.4.1 is proved in Section 3.7.

### 3.5 Digit Hash

The first step to prove Theorem 3.4.1 is to define the class of password mindhashes $\mathcal{G}$. In this section, we introduce Digit mindhash and analyze it's HUM complexity.

**Preprocessing (memorization).** The mindhash needs the user to memorize a single fixed string and a random map from letters to digits, one digit per letter. While this might seem daunting at first, it can in fact be done with 15-30 minutes of effort up front and a total of 1 hour over the human's lifetime (the rest of the hour is for spaced rehearsals, which are well-known to be effective [39, 40, 14]).

More precisely, let $\mathcal{A}$ be the alphabet with ordering $\mathcal{A} = \{a_1, \ldots, a_N\}$ and size $N$. In typical usage, $\mathcal{A} = \{A, B, \ldots, Z\}$ and $N = 26$. Let $\mathcal{S}$ be a set of letters, digits and special

characters such that $|\mathcal{S}| \geq 10$. We denote the set of digits by $[10]$, i.e., $[10] = \{0, \ldots, 9\}$. In the preprocessing phase, the human must

1. Create and memorize a a random map $f : \mathcal{A} \to [10]$ from letters to digits.

2. Memorize a single random string $s = s_1 \ldots s_{K-1} \in \mathcal{S}^{K-1}$.

The purpose of memorizing a fixed string $s$ is two-fold. First, it helps satisfy password restrictions such as "at least one digit, one capitalized letter and one special character" [53]. Second, it makes it harder for the adversary to correctly guess any single password.

**Algorithm 6:** Digit mindhash

**Input**: Challenge $c = c_1 \ldots c_n$, mapping $f$, character string $s$
Compute $g = f(c_1) + \ldots + f(c_n) \pmod{10}$
**Output**: $gs$

**Processing.** For every string $s$ and letter to digit map $f$, let $D_{f,s} : \mathcal{C} \to \mathcal{R}$ be the challenge to response map defined by the Digit hash. The class of password hashes $\mathcal{G}$ is defined as

$$\mathcal{G} = \{D_{f,s} \,|\, f : \mathcal{A} \to [10], s \in \mathcal{S}^{K-1}\}$$

**Human Usability Measure.** Algorithm 7 describes the the processing in the human computation model. This will allow us to analyze its HUM complexity.

For a challenge of length $n$, we have:

$$\text{HUM} = 3 + (n-1)(1 + 1.5 + 1) + 2 + (K-1)(1+1) < 2K + 3.5n$$

This is for a response of length $K$. We can make Digit hash more usable by allowing user to read only the first $i$ letters of the challenge. This will reduce the cost to $2K + 3.5i$. E.g., if $i = 4$ (user only reads the first four letters of the challenge), then we have HUM$= 2K + 14$. In Section 3.9, we will show that this variation of Digit hash still leads to high values of $Q$.

**Algorithm 7:** Digit mindhash HUM

| | |
|---|---|
| **Input**: Challenge $c = c_1 \dots c_n$ | |
| Retrieve challenge $c$. Pointer $\rightarrow c_1$ | Cost = 1 |
| SUM = map $f$ applied to current character | Cost = 1 |
| Shift pointer to next character | Cost = 1 |
| **while** *not end of* $c$ **do** | |
| Apply map $f$ to current character | Cost = 1 |
| Add to SUM $\pmod{10}$ | Cost = 1.5 |
| Shift pointer to next character | Cost = 1 |
| **end** | |
| **Output**: SUM | Cost = 1 |
| Retrieve fixed string $s$. Pointer $\rightarrow s_1$ | Cost = 1 |
| **while** *not end of* $s$ **do** | |
| Output current character | Cost = 1 |
| Shift pointer to next character | Cost = 1 |
| **end** | |

### 3.6 Sample complexity of learning Digit hash from linearly independent challenges

So far we introduced the class of Digit hashes and analyzed it's HUM complexity, this proves the first part of Theorem 3.4.1. In Section 3.6 and 3.7 we prove the second part of Theorem 3.4.1, a lower bound on the sample complexity of learning the class of Digit hashes. Theorem 3.6.1 is the main component of our proof.

Recall that a set of vectors in $\Re^N$ is linearly independent $\pmod{10}$ iff it is linearly independent $\pmod{2}$ and $\pmod{5}$. The fixed string $s$ used in the mindhash is assumed to be uniform among at least $10^{K-1}$ choices.

**Theorem 3.6.1.** *Denote the output of Digit hash on a challenge $C^*$, by $R(C^*)$.*

*a. For any challenge $C_0 \in \mathcal{C}$ and any $r \in [10]$,*

$$\Pr[R(C_0) = rs] \leq \frac{1}{10^K}.$$

*b. For any sequence of $m-1$ challenge-response pairs*

*$\big(C_1, R(C_1)\big), \dots, \big(C_{m-1}, R(C_{m-1})\big)$, and any other challenge $C_m$ s.t.,*

*the set* $\{C_1, \ldots C_m\}$ *is linearly independent* $\pmod{10}$*,*

$$\Pr[R(C_m) = rs \,|\, (C_1, R(C_1)), \ldots, (C_{m-1}, R(C_{m-1})))] = \frac{1}{10}.$$

The next two lemmas provide the essential tools to prove Theorem 3.6.1.

**Lemma 3.6.2.** *Let* $p$ *be a prime number and* $C \in [p]^{k \times N}$ *be a matrix with rank* $k \pmod{p}$*. For any* $g \in [p]^k$*, the number of solutions of* $Cx \equiv g \pmod{p}$ *is exactly* $p^{N-k}$*.*

*Proof.* By assumption, the rows of matrix $C$ are linearly independent $\pmod{p}$, thus there must be $k$ columns that are linearly independent $\pmod{p}$.
Call these columns $\{C^1, \ldots, C^k\}$.

We choose the values of $x_{k+1}, \ldots, x_N$ arbitrarily, each from $[p]$. This is a total of $p^{N-k}$ choices. Consider any one such choice and simplify the equation $Cx \equiv g \pmod{p}$ :

$$[C^1, \ldots, C^k] \begin{bmatrix} x_{j_1} \\ \vdots \\ x_{j_k} \end{bmatrix} \equiv \begin{bmatrix} g'_1 \\ \vdots \\ g'_k \end{bmatrix} \pmod{p}$$

where $g'_l = g_l - \sum_{i=k+1}^{N} C_{li} x_i$ for all $l \in \{1, \ldots, k\}$. Since the matrix $[C^1, \ldots, C^k]$ is full rank $\pmod{p}$, the above system has a unique solution. So we have that for every setting of $x_{k+1}, \ldots, x_N$, there is a unique setting of $x_1, \ldots, x_k$ that satisfies the system $Cx = g$. Hence the total number of solutions to $Cx \equiv g \pmod{p}$ is equal to the number of possible sets $\{x_{k+1}, \ldots, x_N\}$ which is $p^{N-k}$. □

**Lemma 3.6.3.** *Let* $p, q$ *be two distinct prime numbers and* $C \in [pq]^{k \times N}$ *be a matrix with rank* $k \pmod{p}$ *and* $\pmod{q}$*. For any* $g \in [pq]^k$*, the number of solutions to* $Cx \equiv g \pmod{pq}$ *is exactly* $(pq)^{N-k}$*.*

*Proof.* We want to count the number of solutions to $Cx \equiv g \pmod{pq}$. Let $S_p$ to be the set of solutions to $Cx \equiv g \pmod{p}$ and similarly $S_q$ to be the set of solutions to

$Cx \equiv g \pmod{q}$. Using the assumption on $C$ and Lemma 3.6.2, we have $|S_p| = p^{N-k}$ and $|S_q| = q^{N-k}$. We will show that that there is a one-to-one correspondence between the set $\{(x, y) : x \in S_p, y \in S_q\}$ and the set of solutions to $Cx \equiv g \pmod{pq}$.

For any pair $x \in S_p$ and $y \in S_q$, for each coordinate $i \in \{1, \ldots, k\}$, by the Chinese Remainder Theorem applied to the case of 2 primes, there is a unique $z_i \in [pq]$ s.t. $z_i \equiv x_i \pmod{p}$ and $z_i = y_i \pmod{q}$. For the case of two primes, $z_i = x_i q q^* + y_i p p^* \pmod{pq}$ where $q^*$ is the multiplicative inverse of $q \pmod{p}$, i.e., $qq^* \equiv 1 \pmod{p}$ and similarly $pp^* \equiv 1 \pmod{q}$. The mapping $(x_i, y_i) \to z_i$ is thus one-to-one. From this it follows that for each such pair $(x, y)$, there is a unique $z \in [pq]^k$ that satisfies $Cx \equiv g \pmod{pq}$. □

Now we are ready to prove Theorem 3.6.1.

*Proof.* (Theorem 3.6.1)

**Part (a):** By the assumption on $s$, the probability that $s$ is any particular string is at most $1/10^{K-1}$. Thus we only need to compute $\Pr[R(C_0)_1 = r] = \Pr[f(C_{01}) + \ldots + f(C_{0n}) \equiv r \pmod{10}]$ for any $r \in [10]$. To do this, we can count the number of maps $f$ that satisfy

$$f(C_{01}) + \ldots + f(C_{0n}) \equiv r \pmod{10} \tag{3.1}$$

and divide it by the total number of maps $f : \mathcal{A} \to \mathcal{D}$. Using Lemma 3.7.1 with $k = 1$, this probability is exactly $10^{N-1}/10^N$ = 1/10. Thus, the overall probability is at most $1/10^K$.

**Part (b):** Now assume that the adversary has observed $m - 1$ (challenge, response) pairs: $\big(C_1, R(C_1) = g_1 s\big), \ldots, \big(C_{m-1}, R(C_{m-1}) = g_{m-1} s\big)$. For any $r \in [10]$, we want to compute

$$\begin{aligned}
&\Pr[\big(R(C_m) = rs\big) \mid \big(R(C_1) = g_1 s\big), \ldots, \big(R(C_{m-1}) = g_{m-1} s\big)] \\
&= \Pr[\big(R(C_m)_1 = g_m\big) \mid \big(R(C_1)_1 = g_1\big), \ldots, \big(R(C_{m-1})_1 = g_{m-1}\big)] \\
&= \frac{\Pr[\big(R(C_1)_1 = g_1\big), \ldots, \big(R(C_m)_1 = g_m\big)]}{\Pr[\big(R(C_1)_1 = g_1\big), \ldots, \big(R(C_{m-1})_1 = g_{m-1}\big)]}
\end{aligned}$$

We start by computing the value of the numerator. The denominator computation is similar. We need to count the number of mappings $f$ that satisfy

$$\begin{cases} f(C_{11}) + \ldots + f(C_{1n}) \equiv g_1 \pmod{10} \\ \vdots \\ f(C_{m1}) + \ldots + f(C_{mn}) \equiv g_m \pmod{10} \end{cases} \tag{3.2}$$

Lemma 3.7.1 shows that the number of solutions to above $m$ linear equations is $10^{N-m}$. Therefore, the value of the ratio (3.1) is equal to

$$\frac{10^{N-m}}{10^{N-(m-1)}} = \frac{1}{10}.$$

□

### 3.7 Random Challenges

We have seen that as long as a set of random challenges form a linearly independent set $(\mathrm{mod}\ 10)$, the adversary can not predict the response of the Digit hash to a new challenge with probability higher than $1/10$. In this section, we analyze the probability that a set of random challenges is linearly independent $(\mathrm{mod}\ 10)$. This provide us essential tools to prove the main result of this section, Corollary 3.7.2.

Given a set of challenges $\mathcal{C}$, let $p_{i0}$ be the probability that letter $a_i$ does not occur in a random challenge $c \in \mathcal{C}$. In real world applications, we expect $p_0 >> 1/2$ and also the number of repetitions of any letter of the alphabet in a challenge to be less than four. Define $p_0 = \sum_{i=1}^{N} p_{i0}/N$ and $p_1 = 1 - p_0$. For any $n \geq 1$, we model a random challenge matrix $C_{n\times N}$ $(n \geq 1)$ as following. We denote $C_i$ to be the $i^{th}$ row $C$.

◇ *for* i=1:n

(1) Every entry of $C_i$ is chosen independently and identically with $\Pr(C_{ij} = 0) =$

$p_0$ and $\Pr(C_{ij} = l) = p_1/4$ for $1 \leq l \leq 4$.

(2) Repeat step (1) if $C_i = 0_{1\times N}$.

Consider a challenge matrix $C$ generated as above. Lemma 3.7.1 provides a lower bound on the probability that $C$ has full row rank.

**Lemma 3.7.1.** *Let $P^{10}(n)$ be the probability that the challenge matrix $C_{n\times N}$ has full row rank, then*

$$P^{10}(n) \geq \Pi_{j=N-n+1}^{N}(1-(1/2)^j) + \Pi_{j=N-n+1}^{N}(1-(1/5)^j) - 1$$

*Proof.* Let $C^2 := C \pmod 2$ and $P^2(n)$ be the probability that $C^2$ has full row rank $\pmod 2$. Similarly we define $C^5$ and $P^5(n)$. The probability that $C$ has full row rank $\pmod{10}$ is the joint probability that $C^2$ has full row rank $\pmod 2$ and $C^5$ has full row rank $\pmod 5$. By definition of $C$, $C^2$ and $C^5$ are distributed as following:

$$\begin{aligned} &\Pr[C^2_{ij} = 0] = p_0 + p_1/2 \quad &&\Pr[C^5_{ij} = 0] = p_0 \\ &\Pr[C^2_{ij} = 1] = p_1/2 &&\Pr[C^5_{ij} = l] = p_1/4 \;\; 1 \leq l \leq 4 \end{aligned}$$

Theorem 2 of [54] proves that

$$\begin{aligned} \lim_{N\to\infty} \Pr(C^2 \text{ has full row rank} \pmod 2) &= \Pi_{j=N-n+1}^{\infty}(1-(1/2)^j) \\ \lim_{N\to\infty} \Pr(C^5 \text{ has full row rank} \pmod 5) &= \Pi_{j=N-n+1}^{\infty}(1-(1/5)^j) \end{aligned}$$

Although the above values are calculated in the limit, they give good approximation of $P^2(n)$ and $P^5(n)$ for constant values of $N$.

$$\begin{aligned} P^2(n) &\sim \Pi_{j=N-n+1}^{\infty}(1-(1/2)^j) \\ &= \Pi_{j=N+1}^{\infty}(1-(1/2)^j)\Pi_{j=N-n+1}^{N}(1-(1/2)^j) \\ &\simeq \Pi_{j=N-n+1}^{N}(1-(1/2)^j) \end{aligned}$$

Similarly

$$P^5(n) \simeq \Pi_{j=N-n}^{N}(1-(1/5)^j)$$

For large enough values of $N$, we can approximate $\Pi_{j=N+1}^{\infty}(1-(1/2)^j) \simeq 1$. Specifically, this holds for the case of English alphabet where $N = 26$. The final step of the proof, is to apply union bound on the approximations of $P^2(n)$ and $P^5(n)$.

$$\begin{aligned} P^{10}(n) &\geq 1-(1-P^2(n))-(1-P^5(n)) \\ &= P^2(n)+P^5(n)-1 \\ &\simeq \Pi_{j=N-n+1}^{N}(1-(1/2)^j)+\Pi_{j=N-n+1}^{N}(1-(1/5)^j)-1 \end{aligned}$$

□

**Corollary 3.7.2.** *Given an algorithm $A$ that weak $(1/10,\delta)$-PAC learns $\mathcal{G}$ with $m_A(1/10,\delta)$ samples*

$$m_A(1/10,\delta) \geq N - \lceil \log(\frac{2}{1-\delta}) \rceil \quad \forall \delta \in [0,1]$$

*Proof.* Define the function $l(n) := \Pi_{j=N-n+1}^{N}(1-(1/2)^j)+\Pi_{j=N-n+1}^{N}(1-(1/5)^j)-1$. Domain of $l^{-}(\delta)$ can naturally be generalized to the the whole interval $[0,1]$ as following:

$$\begin{cases} \delta \in [l(N+1), l(N)) & l^{-1}(\delta) = N+1 \\ \vdots & \\ \delta \in [l(2), l(1)) & l^{-1}(\delta) = 2 \\ \delta = l(1) & l^{-1}(\delta) = 1 \end{cases}$$

Given $\delta \in [0,1]$, we prove that $l^{-1}(\delta) \geq N - \lceil \log(\frac{2}{1-\delta}) \rceil$. Let $a = \lceil \log \frac{2}{\delta} \rceil$.

$$\begin{aligned} l(N-a) &= \Pi_{j=a+1}^{N}(1-(1/2)^j) + \Pi_{j=a+1}^{N}(1-(1/5)^j) - 1 \\ &\geq 1 - \sum_{j=a+1}^{N}(1/2)^j + 1 - \sum_{j=a+1}^{N}(1/5)^j - 1 \\ &\geq 1 - (1/2)^a - (5/4)(1/5)^a \\ &\geq 1 - \delta \end{aligned}$$

Thus, for any $\delta \in [0,1]$, $l(N - \lceil \log(\frac{2}{1-\delta}) \rceil) \geq \delta$. Since $l$ is a decreasing function $l^{-1}(\delta) \geq N - \lceil \log(\frac{2}{1-\delta}) \rceil$. Assume that an algorithm $A$ weak $(1/10, \delta)$-PAC learns the class $\mathcal{G}$ with $m_A(1/10, \delta)$ samples. Lemma. 3.7.1 states that the probability that $A$ learns $\mathcal{G}$ with $i$ samples is smaller than $1 - l(i)$. In other words, for every $\delta \in [0,1]$, the minimum number of samples that algorithm $A$ needs in order to $(1/10, \delta)$-learns $\mathcal{C}$ is $l^{-1}(\delta) \geq N - \lceil \log(\frac{2}{1-\delta}) \rceil$.

□

*Proof.* (Theorem 3.4.1 part 2) This is a direct consequence of Corollary 3.7.2. Just note that $|\mathcal{G}| = N + (K-1)$

□

### 3.8 Experiments

In order to compute the number of samples that Digit hash requires to weak $(1/10, \delta)$-PAC learn a dictionary of challenge $\mathcal{C}$, we need to answer the following question:

**Question.** Given a dictionary of challenges $\mathcal{C}$, suppose that in each round, a new challenge $c \in \mathcal{C}$ is chosen uniformly at random. What is the expectation of maximum number of rounds up to which the challenges form a linearly independent set $(\mathrm{mod}\ 10)$?

Table 3.1 measures the above expectation over $10K$ trials when dictionaries of challenges are English words, random strings, or the top $400$ most popular website names.

Table 3.1: Expected number of samples required by Digit hash to learn standard dictionaries for $p = 1/10$.

| Dictionary | Number of samples |
|---|---|
| Random $8$-letter strings | 23.2 |
| English dictionary | 20.1 |
| $400$ top websites | 16.6 |

Table 3.2: Expected number of samples required by Digit hash for $p = 1/10$ and different number of letters.

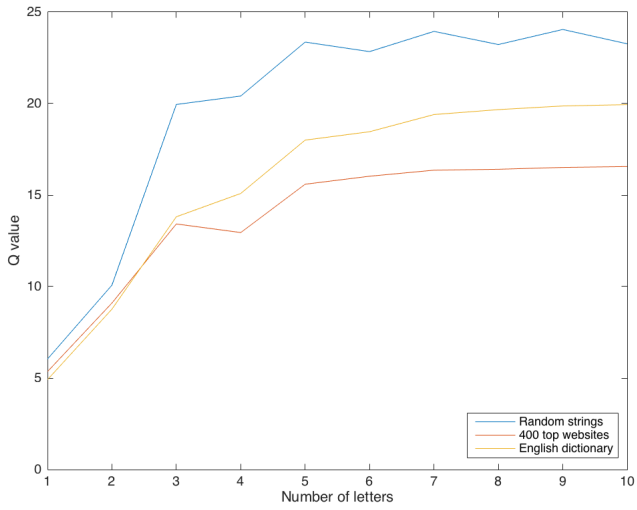
| size | Random strings | English dictionary | 400 top websites |
|---|---|---|---|
| 1 | 6 | 4.9 | 5.3 |
| 2 | 10.1 | 8.8 | 9.1 |
| 3 | 19.9 | 13.8 | 13.4 |
| 4 | 20.4 | 15.1 | 12.9 |
| 5 | 23.4 | 18 | 15.6 |
| 6 | 22.8 | 18.5 | 16 |
| 7 | 23.9 | 19.4 | 16.4 |
| 8 | 23.2 | 19.7 | 16.4 |
| 9 | 24 | 19.9 | 16.5 |
| 10 | 23.3 | 19.9 | 16.6 |

### 3.9 Lower $P$ and adding more initial randomness

There are different variants of Digit hash in terms of security and human usability. The fact that the randomness of one's password is summarized in one digit might be worrisome to some users. On the other hand, some of the users might prefer less amount of computation while generating their passwords. In this section, we introduce different variants of digit hash which satisfy these constraints.

Security of an individual password is inversely proportional to the probability measure $p$ in the password game. The parameter $p$ can be reduced by generating more random digits in each individual password. More precisely, to get probability threshold $p$, each password

Figure 3.1: Expected number of samples required by Digit hash for $p = 1/10$ and different number of letters.




should include $\lceil \log_{10}(1/p) \rceil$ digits. E.g., for $p = 1/1000$ we use 3 digits. Here are different variations of Digit hash which satisfy this goal:

- ◇ Given a challenge $c$, first digit $g_1$ is the same as the original mindhash, the sum of all map values $\pmod{10}$. For the second digit, we use $g_2 = f(c_1) - f(c_2) + f(c_3) - f(c_4) \ldots \pmod{10}$. For the third we use $g_3 = f(c_1) + f(c_2) - f(c_3) - f(c_4) \ldots \pmod{10}$ and so on.

- ◇ First digit $g_1 = f(c_1) + f(c_2) + f(c_3) \pmod{10}$, second digit $g_2 = f(c_4) + f(c_5) + f(c_6) \pmod{10}$ and so on. If length of $c$ is not dividable by 3, wrap around. E.g., for $c = yahoo$, $g_1 = f(y) + f(a) + f(h)$ and $g_2 = f(o) + f(o) + f(y)$. For the dictionary of top 400 websites, the average length of a challenge is 7. Therefore this method will generate 3 digits on average.

Although the above methods provide more security to the individual passwords, observing each password provides more information about the target mindhash to the adversary. Roughly speaking, the second method above scales down the sample complexity by a factor of number of digits.

As we mentioned in Section 3.5, as a variant of Digit hash, we can allow user to only read first $i$ letters of the challenge. This makes the mindhash more humanly usable. Figure 3.1 and Table 3.2 show the expected number of samples that the Digit hash requires to weak $(p, \delta)$-PAC learns standard classes of challenges for $p = 1/10$ and different values of $i$. The expectation is taken over $10K$ trials. In real life, challenges are typically from the set of top website names. In this case, Figure 3.1 shows that, by using only the $5$ first letters of the challenge, the modified Digit hash almost reaches its maximum security.

# CHAPTER 4

# FAIR PCA

We investigate whether the standard dimensionality reduction technique of PCA inadvertently produces data representations with different fidelity for two different populations. We show on several real-world data sets, PCA has higher reconstruction error on population $A$ than on $B$ (for example, women versus men[1] or lower- versus higher-educated individuals). This can happen even when the data set has a similar number of samples from $A$ and $B$. This motivates our study of dimensionality reduction techniques which maintain similar fidelity for $A$ and $B$. We define the notion of Fair PCA and give a polynomial-time algorithm for finding a low dimensional representation of the data which is nearly-optimal with respect to this measure. Finally, we show on real-world data sets that our algorithm can be used to efficiently generate a fair low dimensional representation of the data.

## 4.1 PCA and Representational Bias

In recent years, the ML community has witnessed an onslaught of charges that real-world machine learning algorithms have produced "biased" outcomes. The examples come from diverse and impactful domains. Google Photos labeled African Americans as gorillas [55, 56] and returned queries for CEOs with images overwhelmingly male and white [17], searches for African American names caused the display of arrest record advertisements with higher frequency than searches for white names [19], facial recognition has wildly different accuracy for white men than dark-skinned women [18], and recidivism prediction software has labeled low-risk African Americans as high-risk at higher rates than low-risk white people [16].

The community's work to explain these observations has roughly fallen into either

[1] We use a binary model of gender just as an example and do not mean to exclude other models of gender.

"biased data" or "biased algorithm" bins. In some cases, the training data might under-represent (or over-represent) some group, or have noisier labels for one population than another, or use an imperfect proxy for the prediction label (e.g., using arrest records in lieu of whether a crime was committed). Separately, issues of imbalance and bias might occur due to an algorithm's behavior, such as focusing on accuracy across the entire distribution rather than guaranteeing similar false positive rates across populations, or by improperly accounting for confirmation bias and feedback loops in data collection. If an algorithm fails to distribute loans or bail to a deserving population, the algorithm won't receive additional data showing those people would have paid back the loan, but it will continue to receive more data about the populations it (correctly) believed should receive loans or bail.

Many of the proposed solutions to "biased data" problems amount to re-weighting the training set or adding noise to some of the labels; for "biased algorithms", most work has focused on maximizing accuracy subject to a constraint forbidding (or penalizing) an unfair model. Both of these concerns and approaches have significant merit, but form an incomplete picture of the ML pipeline and where unfairness might be introduced therein. Our work takes another step in fleshing out this picture by analyzing when *dimensionality reduction* might inadvertently introduce bias. We focus on principal component analysis (henceforth PCA), perhaps the most fundamental dimensionality reduction technique in the sciences [21, 23, 22]. We show several real-world data sets for which PCA incurs much higher average reconstruction error for one population than another, even when the populations are of similar sizes. Figure 4.1 shows that PCA on labeled faces in the wild data set (LFW) has higher reconstruction error for women than men even if male and female faces are sampled with equal weight.

This work underlines the importance of considering fairness and bias at every stage of data science, not only in gathering and documenting a data set [57] and in training a model, but also in any interim data processing steps. Many scientific disciplines have adopted PCA as a default preprocessing step, both to avoid the curse of dimensionality and also to

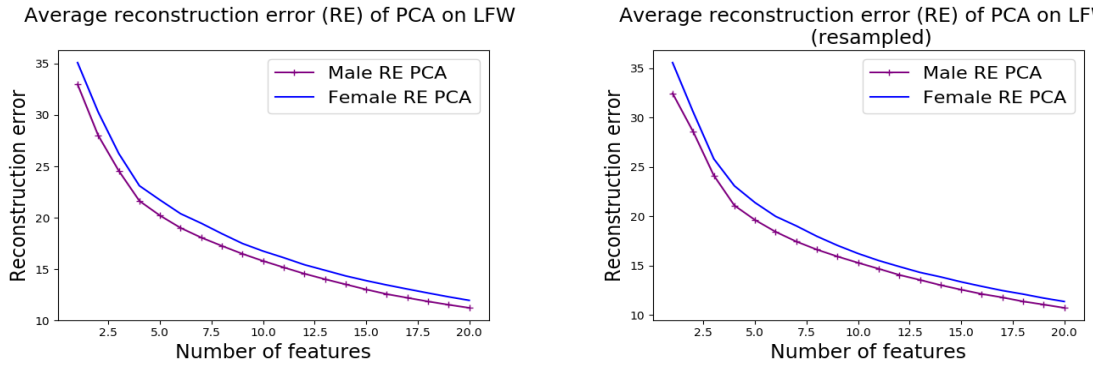



Figure 4.1: Left: Average reconstruction error of PCA on labeled faces in the wild data set (LFW), separated by gender. Right: The same, but sampling 1000 faces with men and women equiprobably (mean over 20 samples).

do exploratory/explanatory data analysis (projecting the data into a number of dimensions that humans can more easily visualize). The study of human biology, disease, and the development of health interventions all face both aforementioned difficulties, as do numerous economic and financial analysis. In such high-stakes settings, where statistical tools will help in making decisions that affect a diverse set of people, we must take particular care to ensure that we share the benefits of data science with a diverse community.

We also emphasize this work has implications for representational rather than just allocative harms, a distinction drawn by [20] between how people are represented and what goods or opportunities they receive. Showing primates in search results for African Americans is repugnant primarily due to its representing and reaffirming a racist painting of African Americans, not because it directly reduces any one person's access to a resource. If the default template for a data set begins with running PCA, and PCA does a better job representing men than women, or white people over minorities, the new representation of the data set itself may rightly be considered an unacceptable sketch of the world it aims to describe.

Our work proposes a different linear dimensionality reduction which aims to represent two populations $A$ and $B$ with similar *fidelity*—which we formalize in terms of *reconstruction error*. Given an $n$-dimensional data set and its $d$-dimensional approximation, the

reconstruction error of the data with respect to its low-dimensional approximation is the sum of squares of distances between the original data points and their approximated points in the $d$-dimensional subspace. To eliminate the effect of size of a population, we focus on average reconstruction error over a population. One possible objective for our goal would find a $d$-dimensional approximation of the data which minimizes the maximum reconstruction error over the two populations. However, this objective doesn't avoid grappling with the fact that population $A$ may perfectly embed into $d$ dimensions, whereas $B$ might require many more dimensions to have low reconstruction error. In such cases, this objective would not necessarily favor a solution with average reconstruction error of $\epsilon$ for $A$ and $y \gg \epsilon$ for $B$ over one with $y$ error for $A$ and $y$ error for $B$. This holds even if $B$ requires $y$ reconstruction error to be embedded into $d$ dimensions and thus the first solution is nearly optimal for both populations in $d$ dimensions.

This motivates our focus on finding a projection which minimizes the maximum *additional* or *marginal* reconstruction error for each population above the optimal $n$ into $d$ projection for that population alone. This quantity captures how much a population's reconstruction error increases by including another population in the dimensionality reduction optimization. Despite this computational problem appearing more difficult than solving "vanilla" PCA, we introduce a polynomial-time algorithm which finds an $n$ into $(d+1)$-dimensional embedding with objective value better than any $d$-dimensional embedding. Furthermore, we show that optimal solutions have equal additional average error for populations $A$ and $B$.

**Summary of our results** We show PCA can overemphasize the reconstruction error for one population over another (equally sized) population, and we should therefore think carefully about dimensionality reduction in domains where we care about fair treatment of different populations. We propose a new dimensionality reduction problem which focuses on representing $A$ and $B$ with similar additional error over projecting $A$ or $B$ individually.

We give a polynomial-time algorithm which finds near-optimal solutions to this problem. Our algorithm relies on solving a semidefinite program (SDP), which can be prohibitively slow for practical applications. We note that it is possible to (approximately) solve an SDP with a much faster multiplicative-weights style algorithm, whose running time in practice is equivalent to solving standard PCA at most 10-15 times. We then evaluate the empirical performance of this algorithm on several human-centric data sets.

### 4.2 Related work

This work contributes to the area of fairness for machine learning models, algorithms, and data representations. One interpretation of our work is that we suggest using Fair PCA, rather than PCA, when creating a lower-dimensional representation of a data set for further analysis. Both pieces of work which are most relevant to our work take the posture of explicitly trying to reduce the correlation between a sensitive attribute (such as race or gender) and the new representation of the data. The first piece is a broad line of work [58, 59, 60, 61, 62] that aims to design representations which will be conditionally independent of the protected attribute, while retaining as much information as possible (and particularly task-relevant information for some fixed classification task). The second piece is the work by [63], who also look to design PCA-like maps which reduce the projected data's dependence on a sensitive attribute. Our work has a qualitatively different goal: we aim not to hide a sensitive attribute, but instead to maintain as much information about each population after projecting the data. In other words, we look for representation with similar richness for population $A$ as $B$, rather than making $A$ and $B$ indistinguishable.

Other work has developed techniques to obfuscate a sensitive attribute directly [64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75]. This line of work diverges from ours in two ways. First, these works focus on representations which obfuscate the sensitive attribute rather than a representation with high fidelity regardless of the sensitive attribute. Second, most of these works do not give formal guarantees on how much an objective will degrade after

their transformations. Our work directly minimizes the amount by which each group's marginal reconstruction error increases.

Much of the other work on fairness for learning algorithms focuses on fairness in classification or scoring [76, 77, 78, 79], or online learning settings [80, 81, 82, 83]. These works focus on either statistical parity of the decision rule, or equality of false positives or negatives, or an algorithm with a fair decision rule. All of these notions are driven by a single learning task rather than a generic transformation of a data set, while our work focuses on a ubiquitous, task-agnostic preprocessing step.

## 4.3 Notation and vanilla PCA

We are given $n$-dimensional data points represented as rows of matrix $M \in \mathbb{R}^{m \times n}$. We will refer to the *set* and *matrix* representation interchangeably. The data consists of two subpopulations $A$ and $B$ corresponding to two groups with different value of a binary sensitive attribute (e.g., males and females). We denote by $\begin{bmatrix} A \\ B \end{bmatrix}$ the concatenation of two matrices $A, B$ by row. We refer to the $i^{th}$ row of $M$ as $M_i$, the $j^{th}$ column of $M$ as $M^j$ and the $(i,j)^{th}$ element of $M$ as $M_{ij}$. We denote the Frobenius norm of matrix $M$ by $\|M\|_F$ and the 2-norm of the vector $M_i$ by $\|M_i\|$. For $k \in \mathbb{N}$, we write $[k] := \{1, \ldots, k\}$. $|A|$ denotes the size of a set $A$. Given two matrices $M$ and $N$ of the same size, the Frobenius inner product of these matrices is defined as $\langle M, N \rangle = \sum_{ij} M_{ij} N_{ij} = \mathrm{Tr}(M^T N)$.

### 4.3.1 PCA

This section recalls useful facts about PCA that we use in later sections. We begin with a reminder of the definition of the PCA problem in terms of minimizing the reconstruction error of a data set.

**Definition 4.3.1.** *(PCA problem) Given a matrix $M \in \mathbb{R}^{m \times n}$, find a matrix $\widehat{M} \in \mathbb{R}^{m \times n}$ of rank at most $d$ $(d \leq n)$ that minimizes $\|M - \widehat{M}\|_F$.*

We will refer to $\widehat{M}$ as an optimal rank-$d$ approximation of $M$. The following well-known fact characterizes the solutions to this classic problem [e.g., [84]].

**Fact 4.3.1.** *If $\widehat{M}$ is a solution to the PCA problem, then $\widehat{M} = MWW^T$ for a matrix $W \in \mathbb{R}^{n\times d}$ with $W^TW = I$. The columns of $W$ are eigenvectors corresponding to top $d$ eigenvalues of $M^TM$.*

The matrix $WW^T \in \mathbb{R}^{n\times n}$ is called a projection matrix.

## 4.4 Fair PCA

Given the $n$-dimensional data with two subgroups $A$ and $B$, let $\widehat{M}, \widehat{A}, \widehat{B}$ be optimal rank-$d$ PCA approximations for $M, A,$ and $B$, respectively. We introduce our approach to fair dimensionality reduction by giving two compelling examples of settings where dimensionality reduction inherently makes a tradeoff between groups $A$ and $B$. Figure 4.2 shows a setting where projecting onto any single dimension either favors $A$ or $B$ (or incurs significant reconstruction error for both), while either group separately would have a high-fidelity embedding into a single dimension. This example suggests any projection will necessarily make a trade off between error on $A$ and error on $B$.

Our second example (shown in Figure 4.3) exhibits a setting where $A$ and $B$ suffer very different reconstruction error when projected onto one dimension: $A$ has high reconstruction error for every projection while $B$ has a perfect representation in the horizontal direction. Thus, asking for a projection which minimizes the maximum reconstruction error for groups $A$ and $B$ might require incurring additional error for $B$ while not improving the error for $A$. So, minimizing the maximum reconstruction error over $A$ and $B$ fails to account for the fact that two populations might have wildly different representation error when embedded into $d$ dimensions. Optimal solutions to such objective might behave in a counter intuitive way, preferring to exactly optimize for the group with larger inherent representation error rather than approximately optimizing for both groups simultaneously.

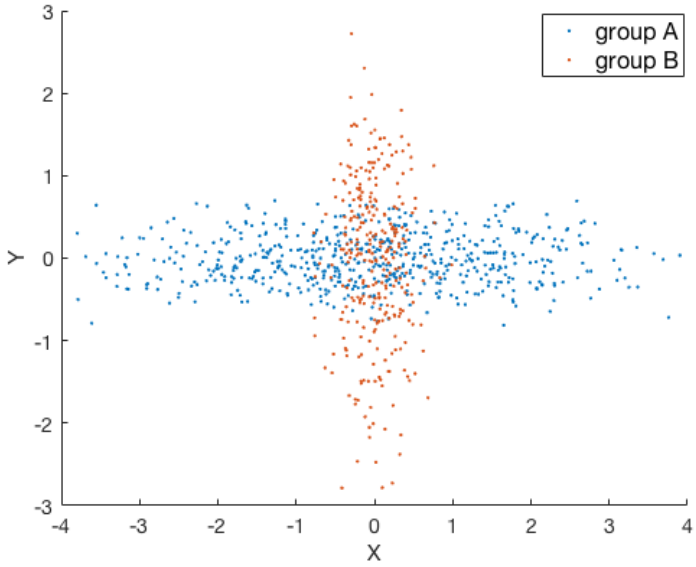



Figure 4.2: The best one dimensional PCA projection for group $A$ is vector $(1, 0)$ and for group $B$ it is vector $(0, 1)$.

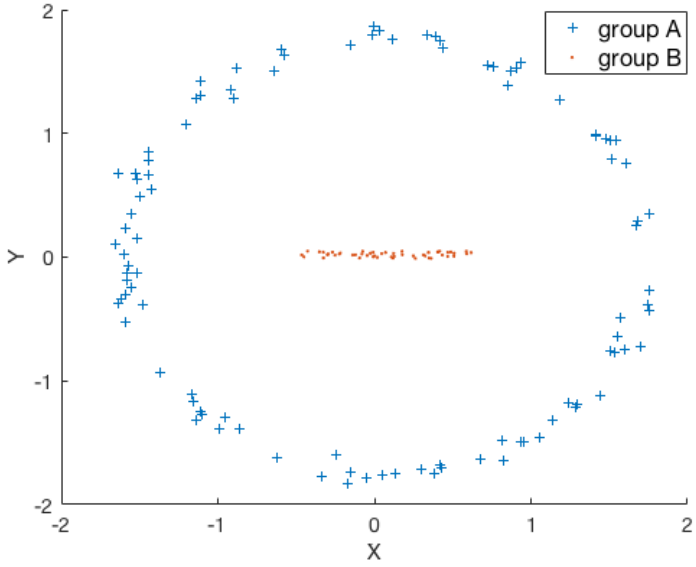



Figure 4.3: Group $B$ has a perfect one-dimensional projection. For group $A$, any one-dimensional projection is equally bad.

We find this behaviour undesirable—it requires sacrifice in quality for one group for no improvement for the other group.

**Remark 4.4.1.** *We focus on the setting where we ask for a single projection into $d$ dimen-*

*sions rather than two separate projections because using two distinct projections (or more generally two models) for different populations raises legal and ethical concerns. Learning two different projections also faces no inherent tradeoff in representing $A$ or $B$ with those projections.*[2]

We therefore turn to finding a projection which minimizes the maximum deviation of each group from its optimal projection. This optimization asks that $A$ and $B$ suffer a similar *loss* for being projected together into $d$ dimensions compared to their individually optimal projections. We now introduce our notation for measuring a group's loss when being projected to $Z$ rather than to its optimal $d$-dimensional representation:

**Definition 4.4.2** (Reconstruction error)**.** *Given two matrices $Y$ and $Z$ of the same size, the reconstruction error of $Y$ with respect to $Z$ is defined as*

$$\textit{error}(Y, Z) = \|Y - Z\|_F^2.$$

**Definition 4.4.3** (Reconstruction loss)**.** *Given a matrix $Y \in \mathbb{R}^{a\times n}$, let $\widehat{Y} \in \mathbb{R}^{a\times n}$ be the optimal rank-$d$ approximation of $Y$. For a matrix $Z \in \mathbb{R}^{a\times n}$ with rank at most $d$ we define*

$$loss(Y, Z) := \|Y - Z\|_F^2 - \|Y - \widehat{Y}\|_F^2.$$

Then, the optimization that we study asks to minimize the maximum loss suffered by any group. This captures the idea that, fixing a feasible solution, the objective will only improve if it improves the loss for the group whose current representation is worse. Furthermore, considering the reconstruction loss and not the reconstruction error prevents the optimization from incurring error for one subpopulation without improving the error for the other one as described in Figure 4.3.

[2] [85] has asked whether equal treatment requires different models for two groups.

**Definition 4.4.4** (Fair PCA)**.** *Given $m$ data points in $\mathbb{R}^n$ with subgroups $A$ and $B$, we define the problem of finding a fair PCA projection into $d$-dimensions as optimizing*

$$\min_{U \in \mathbb{R}^{m \times n},\, \text{rank}(U) \leq d} \max \left\{ \frac{1}{|A|} loss(A, U_A), \frac{1}{|B|} loss(B, U_B) \right\}, \tag{4.1}$$

*where $U_A$ and $U_B$ are matrices with rows corresponding to rows of $U$ for groups $A$ and $B$ respectively.*

This definition does not appear to have a closed-form solution (unlike vanilla PCA—see Fact 4.3.1). To take a step in characterizing solutions to this optimization, Theorem 4.4.5 states that a fair PCA low dimensional approximation of the data results in the same loss for both groups.

**Theorem 4.4.5.** *Let $U$ be a solution to the Fair PCA problem* (4.1)*, then*

$$\frac{1}{|A|} loss(A, U_A) = \frac{1}{|B|} loss(B, U_B).$$

Before proving Theorem 4.4.5, we need to state some building blocks of the proof, Lemmas 4.4.6, 4.4.7, and 4.4.8.

**Lemma 4.4.6.** *Given a matrix $U \in \mathbb{R}^{m \times n}$ such that* rank$(U) \leq d$ *,*
*let $f(U) = \max \left\{ \frac{1}{|A|} loss(A, U_A), \frac{1}{|B|} loss(B, U_B) \right\}$. Let $\{v_1, \ldots, v_d\} \subset \mathbb{R}^n$ be an orthonormal basis of the row space of $U$ and $V := [v_1, \ldots, v_d] \in \mathbb{R}^{n \times d}$. Then*

$$f\left( \begin{bmatrix} A \\ B \end{bmatrix} VV^T \right) = f\left( \begin{bmatrix} AVV^T \\ BVV^T \end{bmatrix} \right) \leq f(U).$$

*Proof.* Since rank$(U) \leq d$, rank$(V) \leq d$ and thus rank$(\begin{bmatrix} A \\ B \end{bmatrix} VV^T) \leq d$. We will first show that $loss(A, AVV^T) \leq loss(A, U_A)$.

**Step 1.** Since $\{v_1, \ldots, v_d\}$ is an orthonormal basis of row space of $U$, for every row of $U_A$, we have that $(U_A)_i = c_i V^T$ for some $c_i \in \mathbb{R}^{1\times d}$.

**Step 2.** We show if we $c_i \rightarrow A_i V$ and consequently substitute the row $(U_A)_i \rightarrow A_i V V^T$, the value of $\|A_i - (U_A)_i\|$ decreases.

$$\|A_i - (U_A)_i\|^2 = \|A_i - c_i V^T\|^2 = A_i {A_i}^T - 2A_i V {c_i}^T + c_i {c_i}^T$$

We used the fact that $V^T V = I$. Minimizing the RHS with respect to $c_i$ results in $c_i = A_i V$.

**Step 3.** Step 2 proved that for every $i$, $\|A_i - A_i V V^T\|^2 \leq \|A_i - (U_A)_i\|^2$. Remember that

$$loss(A, U_A) = \|A - U_A\|_F^2 - \|A - \widehat{A}\|_F^2 = \sum \|A_i - (U_A)_i\|^2 - \|A - \widehat{A}\|_F^2$$
$$loss(A, AVV^T) = \|A - AVV^T\|_F^2 - \|A - \widehat{A}\|_F^2 = \sum \|A_i - A_i VV^T\|^2 - \|A - \widehat{A}\|_F^2$$

This finished the proof that $loss(A, AVV^T) \leq loss(A, U_A)$. Similarly, we can see that $loss(B, BVV^T) \leq loss(B, U_B)$. Therefore

$$f(\begin{bmatrix} A \\ B \end{bmatrix} VV^T) = \max\big(\frac{1}{|A|} loss(A, AVV^T), \frac{1}{|B|} loss(B, BVV^T)\big)$$
$$\leq \max\big(\frac{1}{|A|} loss(A, U_A), \frac{1}{|B|} loss(B, U_B)\big) = f(U)$$

□

The next lemma presents some equalities that we will use frequently in the proofs.

**Lemma 4.4.7.** *Given a matrix $V = [v_1, \ldots, v_d] \in \mathbb{R}^{n\times d}$ with orthonormal columns, we have:*

- $loss(A, AVV^T) = \|\widehat{A}\|_F^2 - \sum_{i=1}^d \|Av_i\|^2 = \|\widehat{A}\|_F^2 - \langle A^T A, VV^T \rangle$

$\diamond$ $\|A - AVV^T\|_F^2 = \|A\|_F^2 - \|AV\|_F^2 = \|A\|_F^2 - \sum_{i=1}^{d} \|Av_i\|^2$

*Proof.* From Lemma 4.3.1, we know that there exist a matrix $W_A \in \mathbb{R}^{n \times d}$ such that $W_A^T W_A = I$ and $\widehat{A} = AW_A W_A^T$. Considering this and the fact that $V^T V = I$

$$
\begin{aligned}
loss(A, AVV^T) &= \|A - AVV^T\|_F^2 - \|A - AW_A W_A^T\|_F^2 \\
&= \sum_i \|A_i - A_i VV^T\|^2 - \|A_i - A_i W_A W_A^T\|^2 \\
&= \sum_i A_i A_i^T - A_i VV^T A_i^T - (\sum_i A_i A_i^T - \sum_i A_i W_A W_A^T) \\
&= \sum_i A_i W_A W_A^T A_i^T - \sum_i A_i VV^T A_i^T \\
\sum_i A_i W_A W_A^T A_i^T &= \sum_i \|A_i W_A\|^2 = \|AW_A\|_F^2 = \|AW_A W_A^T\|_F^2 = \|\widehat{A}\|_F^2 \\
\sum_i A_i VV^T A_i^T &= \sum_i \|A_i V\|^2 = \|AV\|_F^2 = \sum_i \|Av_i\|^2 \\
\sum_i A_i VV^T A_i^T &= \sum_i \|A_i V\|^2 = \|AV\|_F^2 = \mathrm{Tr}(V^T A^T AV) \\
&= \mathrm{Tr}(VV^T A^T A) = \langle A^T A, VV^T \rangle
\end{aligned}
$$

Therefore $loss(A, AVV^T) = \|\widehat{A}\|_F^2 - \sum_{i=1}^{d} \|Av_i\|^2 = \|\widehat{A}\|_F^2 - \langle A^T A, VV^T \rangle$.

$$
\begin{aligned}
\|A - AVV^T\|_F^2 &= \sum_i \|A_i - A_i VV^T\|^2 = \sum_i A_i A_i^T - \sum_i A_i VV^T A_i^T \\
&= \|A\|_F^2 - \sum_i \|Av_i\|^2 = \|A\|_F^2 - \|AV\|_F^2
\end{aligned}
$$

□

Let the function $g_A = g_A(U)$ measure the reconstruction error of a fixed matrix $A$ with respect to its orthogonal projection to the input subspace $U$. The next lemma shows that

the value of the function $g_A$ at any local minimum is the same.

**Lemma 4.4.8.** *Given a matrix $A \in \mathbb{R}^{a \times n}$, and a $d$-dimensional subspace $U$, let the function $g_A = g_A(U)$ denote the reconstruction error of matrix $A$ with respect to its orthogonal projection to the subspace $U$, that is $g_A(U) := \|A - AUU^T\|_F^2$, where by abuse of notation we use $U$ inside the norm to denote the matrix which has an orthonormal basis of the subspace $U$ as its columns. The value of the function $g_A$ at any local minimum is the same.*

*Proof.* We prove that the value of function $g_A$ at its local minima is equal to its value at its global minimum, which we know is the subspace spanned by a top $d$ eigenvectors of $A^TA$. More precisely, we prove: Let $\{v_1, \ldots, v_n\}$ be an orthonormal basis of eigenvectors of $A^TA$ with corresponding eigenvalues $\lambda_1 \geq \ldots \geq \lambda_n$ where ties are broken arbitrarily. Let $V^*$ be the subspace spanned by $\{v_1, \ldots, v_d\}$ and let $U$ be some $d$-dimensional subspace s.t. $g_A(U) > g_A(V^*)$. There is a continuous path from $U$ to $V^*$ s.t. the value of $g_A$ monotonically decreases for every $d$-dimensional subspace on the path.

Before starting the proof, we will make a couple of notes which would be used throughout the proof. First note that $g_A(V)$ is well-defined i.e., the value of $g_A(V)$ is only a function of the subspace $V$. More precisely, $g_A(V)$ is invariant with respect to different choices of orthonormal basis of the subspace $V$. Second, given Lemma 4.4.7, $g_A(V) = \|A\|_F^2 - \sum_i \|Av_i\|^2$. Therefore, proving that $g_A(V)$ is decreasing is equivalent to proving that $\sum_i \|Av_i\|^2$ is increasing as a function of any choice of orthonormal basis of the subspaces on the path.

$g_A(U) > g_A(V^*)$ therefore $U \neq V^*$. Let $k$ be the smallest index such that $v_k \notin U$. Extend $\{v_1, \ldots, v_{k-1}\}$ to an orthonormal basis of $U$: $\{v_1, \ldots, v_{k-1}, v'_k, \ldots, v'_d\}$. Let $q \geq k$ be the smallest index such that $\|Av_q\|^2 > \|Av'_q\|^2$. Such an index $q$ must exist given that $g_A(U) > g_A(V^*)$. Without loss of generality we can assume that $q = 1$. Therefore, we assume that $v_1$, the top eigenvector of $A^TA$, is not in $U$ and that it strictly maximizes the function $\|Au\|^2$ over the space of unit vectors $u$. Specifically, for any unit vector $u \in U$, $\|Au\|^2 < \|Av_1\|^2 = \lambda_1$. Let $v_1 = \sqrt{1-a^2}z_1 + az_2$ where $z_1 \in U$ and $z_2 \perp U$, $\|z_1\| =$

$\|z_2\| = 1$ i.e., the projection of $v_1$ to $U$ is $\sqrt{1-a^2}z_1$. We distinguish two cases:

**Case $z_1 = 0$.** $v_1 \perp U$. Let $w = \sqrt{1-\epsilon^2}u_1 + \epsilon v_1$. $\|w\| = 1$. Note that $\{w, u_2, \ldots, u_d\}$ is an orthonormal set of vectors. We set $U_\epsilon = span\{w, u_2, \ldots, u_d\}$. We show that $g_A(U_\epsilon) < g_A(U)$. Using the formulation of $g$ from Lemma 4.4.7, we need to show that $\|Aw\|^2 + \|Au_2\|^2 + \ldots + \|Au_d\|^2 > \|Au_1\|^2 + \|Au_2\|^2 + \ldots + \|Au_d\|^2$ or equivalently that $\|Aw\|^2 > \|Au_1\|^2$.

$$\begin{aligned}
\|Aw\|^2 - \|Au_1\|^2 &= \|A(\sqrt{1-\epsilon^2}u_1 + \epsilon v_1)\|^2 - \|Au_1\|^2 \\
&= (\sqrt{1-\epsilon^2}u_1^T + \epsilon v_1^T)A^TA(\sqrt{1-\epsilon^2}u_1 + \epsilon v_1) - \|Au_1\|^2 \\
&= (1-\epsilon^2)u_1^TA^TAu_1 + \epsilon^2 v_1^TA^TAv_1 + 2\sqrt{1-\epsilon^2}\epsilon u_1^TA^TAv_1 - \|Au_1\|^2 \\
&= (1-\epsilon^2)\|Au_1\|^2 + \epsilon^2\lambda_1 + 2\epsilon\sqrt{1-\epsilon^2}u_1^TA^TAv_1 - \|Au_1\|^2 \\
&= \epsilon^2(\lambda_1 - \|Au_1\|^2) + 2\epsilon\sqrt{1-\epsilon^2}u_1^TA^TAv_1
\end{aligned}$$

where $u_1^TA^TAv_1 = u_1^T(\lambda_1 v_1) = \lambda_1 u_1^T v_1 = 0$ since $v_1$ is an eigenvector of $A^TA$ and $v_1 \perp u_1$. This, and considering the fact that $\|Au_1\|^2 < \lambda_1$

$$\|Aw\|^2 - \|Au_1\|^2 = \epsilon^2(\lambda_1 - \|Au_1\|^2) > 0$$

Therefore, $\|Aw\|^2 > \|Au_1\|^2$ and thus $g_A(U_\epsilon) < g_A(U)$.

**Case $z_1 \neq 0$.** Note that $z_2 \neq 0$ either since we picked $v_1 \notin U$. Let's extend $\{z_1\}$ to an orthonormal basis of $U$: $\{z_1, u_2, \ldots, u_k\}$. We will transform $U$ s.t. the resulting subspace $U_1$ is the span of $v_1, u_2, \ldots, u_k$. This can then be repeated orthogonal to $v_1$ till the subspace becomes $V^*$.

For small enough $\epsilon > 0$, consider the unit vector $w = \sqrt{1-\epsilon^2}z_1 + \epsilon z_2$. We will move

$U$ to $U_\epsilon := span\{w, u_2, \ldots, u_d\}$. The latter is an orthonormal representation since both $z_1$ and $z_2$ are orthogonal to all of $u_2, \ldots, u_d$ and $w$ is in the span of $z_1, z_2$. We will prove that $g_A(U_\epsilon) < g_A(U)$. Given Lemma 4.4.7, since the chosen orthonormal basis of these two subspaces differ only in $w$ and $z_1$, it suffices to show that $\|Aw\|^2 > \|Az_1\|^2$. We can write

$$\begin{aligned} w &= \left(\sqrt{1-\epsilon^2} - \frac{\epsilon\sqrt{1-a^2}}{a}\right) z_1 + \frac{\epsilon}{a}\left(\sqrt{1-a^2}z_1 + az_2\right) \\ &= \left(\sqrt{1-\epsilon^2} - \frac{\epsilon\sqrt{1-a^2}}{a}\right) z_1 + \frac{\epsilon}{a}v_1. \end{aligned}$$

Thus, noting that $A^TAv_1 = \lambda_1 v_1$ ($v_1$ is an eigenvector with eigenvalue $\lambda_1$) and $z_1^T v_1 = \sqrt{1-a^2}$,

$$\begin{aligned} &\|Aw\|^2 = \\ &\left(\sqrt{1-\epsilon^2} - \frac{\epsilon\sqrt{1-a^2}}{a}\right)^2 \|Az_1\|^2 + \frac{\epsilon^2}{a^2}\|Av_1\|^2 + 2\frac{\epsilon}{a}\left(\sqrt{1-\epsilon^2} - \frac{\epsilon\sqrt{1-a^2}}{a}\right) z_1^T A^T A v_1 \\ &= \left(1 - \epsilon^2 + \frac{\epsilon^2(1-a^2)}{a^2} - 2\frac{\epsilon\sqrt{(1-\epsilon^2)(1-a^2)}}{a}\right)\|Az_1\|^2 + \frac{\epsilon^2}{a^2}\lambda_1 \\ &+ 2\frac{\epsilon}{a}\left(\sqrt{1-\epsilon^2} - \frac{\epsilon\sqrt{1-a^2}}{a}\right)\lambda_1 z_1^T v_1 \\ &= \left(1 - 2\epsilon^2 + \frac{\epsilon^2}{a^2} - 2\frac{\epsilon\sqrt{(1-\epsilon^2)(1-a^2)}}{a}\right)\|Az_1\|^2 \\ &+ \left(\frac{\epsilon^2}{a^2} + 2\frac{\epsilon\sqrt{(1-\epsilon^2)(1-a^2)}}{a} - 2\frac{\epsilon^2(1-a^2)}{a^2}\right)\lambda_1 \\ &= \|Az_1\|^2 + (\lambda_1 - \|Az_1\|^2)\left(2\frac{\epsilon\sqrt{(1-\epsilon^2)(1-a^2)}}{a} + 2\epsilon^2 - \frac{\epsilon^2}{a^2}\right) > \|Az_1\|^2 \end{aligned}$$

The last inequality follows since $\lambda_1 > \|Az_1\|^2$ and we can choose $0 < \epsilon < \frac{1}{1+C}$ for $C = 4a^2(1-a^2)$ so that $2\frac{\epsilon\sqrt{(1-\epsilon^2)(1-a^2)}}{a} > \frac{\epsilon^2}{a^2}$ . Thus, $\|Aw\|^2 > \|Az_1\|^2$ and therefore $g_A(U_\epsilon) < g_A(U)$. □

**Proof of Theorem 4.4.5**:

Consider the functions $g_A$ and $g_B$ defined in Lemma 4.4.8. It follows from Lemma 4.4.6

and Lemma 4.4.7 that for $V \in \mathbb{R}^{n\times d}$ with $V^TV = I$ we have

$$loss(A, AVV^T) = \|\widehat{A}\|_F^2 - \|A\|_F^2 + g_A(V), \tag{4.2}$$
$$loss(B, BVV^T) = \|\widehat{B}\|_F^2 - \|B\|_F^2 + g_B(V).$$

Therefore, the Fair PCA problem is equivalent to

$$\min_{V\in\mathbb{R}^{n\times d}, V^TV=I} f(V) := \max\left\{\frac{1}{|A|}loss(A, AVV^T), \frac{1}{|B|}loss(B, BVV^T)\right\}.$$

We proceed to prove the claim by contradiction. Let $W$ be a global minimum of $f$ and assume that

$$\frac{1}{|A|}loss(A, AWW^T) > \frac{1}{|B|}loss(B, BWW^T). \tag{4.3}$$

Hence, since $loss$ is continuous, for any matrix $W_\epsilon$ with $W_\epsilon^TW_\epsilon = I$ in a small enough neighborhood of $W$, $f(W_\epsilon) = \frac{1}{|A|}loss(A, AW_\epsilon W_\epsilon^T)$. Since $W$ is a global minimum of $f$, it is a local minimum of $\frac{1}{|A|}loss(A, AWW^T)$ or equivalently a local minimum of $g_A$ because of (4.2).

Let $\{v_1, \ldots, v_n\}$ be an orthonormal basis of the eigenvectors of $A^TA$ corresponding to eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$. Let $V^*$ be the subspace spanned by $\{v_1, \ldots, v_d\}$. Note that $loss(A, AV^{*T}V^*) = 0$. Since the loss is always non-negative for both $A$ and $B$, (4.3) implies that $loss(A, AWW^T) > 0$. Therefore, $W \neq V^*$ and $g_A(V^*) < g_A(W)$. By Lemma 4.4.8, this is in contradiction with $V^*$ being a global minimum and $W$ being a local minimum of $g_A$. □

## 4.5 Algorithm and analysis

In this section, we present a polynomial-time algorithm for solving the fair PCA problem. Our algorithm outputs a matrix of rank at most $d + 1$ and guarantees that it achieves the

fair PCA objective value equal to the optimal $d$-dimensional fair PCA value. The algorithm has two steps: first, relax fair PCA to a semidefinite optimization problem and solve the SDP; second, solve an LP designed to reduce the rank of said solution. We argue using properties of extreme point solutions that the solution must satisfy a number of constraints of the LP with equality, and argue directly that this implies the solution must lie in $d+1$ or fewer dimensions. We refer the reader to [86] for basics and applications of this technique in approximation algorithms.

**Theorem 4.5.1.** *There is a polynomial-time algorithm that outputs an approximation matrix of the data such that it is either of rank $d$ and is an optimal solution to the fair PCA problem OR it is of rank $d+1$, has equal losses for the two populations and achieves the optimal fair PCA objective value for dimension $d$.*

*Proof.* The algorithm to prove Theorem 4.5.1 is presented in Algorithm 8.
Using Lemma 4.4.7, we can write the semi-definite relaxation of the fair PCA objective (Def. 4.4.4) as SDP (4.4). This semi-definite program can be solved in polynomial time. The system of constraints (4.5)-(4.11) is a linear program in the variables $\lambda_i$ (with the $u_i$'s fixed). Therefore, an extreme point solution $(\bar{\lambda}, z^*)$ is defined by $n+1$ equalities, at most three of which can be constraints in (4.7)-(4.10) and the rest (at least $n-2$ of them) must be from the $\bar{\lambda}_i = 0$ or $\bar{\lambda}_i = 1$ for $i \in [n]$. Given the upper bound of $d$ on the sum of the $\bar{\lambda}_i$'s, this implies that at least $d-1$ of them are equal to 1, i.e., at most two are fractional and add up to 1.

**Case 1.** All the eigenvalues are integral. Therefore, there are $d$ eigenvalues equal to 1. This results in orthogonal projection to $d$-dimension.

**Case 2.** $n-2$ of eigenvalues are in $\{0,1\}$ and two eigenvalues $0 < \bar{\lambda}_d, \bar{\lambda}_{d+1} < 1$. Since we have $n+1$ tight constraints, this means that both of the first two constraints are tight. Therefore

**Algorithm 8:** Fair PCA

**Input :** $A \in \mathbb{R}^{m_1 \times n}, B \in \mathbb{R}^{m_2 \times n}, d < n, m = m_1 + m_2$

**Output:** $U \in \mathbb{R}^{m \times n}, \text{rank}(U) \leq d + 1$

Find optimal rank-$d$ approximations of $A, B$ as $\widehat{A}, \widehat{B}$ (e.g. by Singular Value Decomposition)

Let $(\hat{P}, \hat{z})$ be a solution to the SDP:

$$\begin{aligned} \min_{P \in \mathbb{R}^{n \times n},\, z \in \mathbb{R}} \quad & z && (4.4)\\ \text{s.t. } z &\geq \frac{1}{m_1} \cdot \left( \|\widehat{A}\|_F^2 - \langle A^\top A, P \rangle \right) \\ z &\geq \frac{1}{m_2} \cdot \left( \|\widehat{B}\|_F^2 - \langle B^\top B, P \rangle \right) \\ & \text{Tr}(P) \leq d,\ 0 \preceq P \preceq I \end{aligned}$$

Apply Singular Value Decomposition to $\hat{P}$, $\hat{P} = \sum_{j=1}^n \hat{\lambda}_j u_j u_j^T$

Find an extreme solution $(\bar{\lambda}, z^*)$ of the LP:

$$\begin{aligned} \min_{\lambda \in \mathbb{R}^n,\, z \in \mathbb{R}} \quad & z && (4.5)\\ \text{s.t. } z &\geq \frac{1}{m_1}(\|\widehat{A}\|_F^2 - \langle A^\top A, \textstyle\sum_{j=1}^n \lambda_j u_j u_j^T \rangle) && (4.6)\\ &= \frac{1}{m_1}(\|\widehat{A}\|_F^2 - \textstyle\sum_{j=1}^n \lambda_j \cdot \langle A^\top A, u_j u_j^T \rangle) && (4.7)\\ z &\geq \frac{1}{m_2}(\|\widehat{B}\|_F^2 - \langle B^\top B, \textstyle\sum_{j=1}^n \lambda_j u_j u_j^T \rangle) && (4.8)\\ &= \frac{1}{m_2}(\|\widehat{B}\|_F^2 - \textstyle\sum_{j=1}^n \lambda_j \cdot \langle B^\top B, u_j u_j^T \rangle) && (4.9)\\ & \textstyle\sum_{i=1}^n \lambda_i \leq d && (4.10)\\ & 0 \leq \lambda_i \leq 1 && (4.11) \end{aligned}$$

Set $P^* = \sum_{j=1}^n \lambda_j^* u_j u_j^T$ where $\lambda_j^* = 1 - \sqrt{1 - \bar{\lambda}_j}$.

**return** $U = \begin{bmatrix} A \\ B \end{bmatrix} P^*$

$$\frac{1}{|A|}(\|\widehat{A}\|_F^2 - \sum_{i=1}^{n} \bar{\lambda}_i \langle A^T A, u_i u_i^T \rangle) = \frac{1}{|B|}(\|\widehat{B}\|_F^2 - \sum_{i=1}^{n} \bar{\lambda}_i \langle B^T B, u_i u_i^T \rangle) = z^* \leq \hat{z},$$

where the inequality is by observing that $(\hat{\lambda}, \hat{z})$ is a feasible solution. Note that the loss of group $A$ given by an affine projection $P^* = \sum_{j=1}^{n} \lambda^* u_j u_j^T$ is

$$\begin{aligned}
&loss(A, AP^*) \\
&= \|A - AP^*\|_F^2 - \|A - \widehat{A}\|_F^2 = \text{Tr}\left((A - AP^*)(A - AP^*)^\top\right) - \|A\|_F^2 + \|\widehat{A}\|_F^2 \\
&= \text{Tr}\left((A - AP^*)(A - AP^*)^\top\right) - \|A\|_F^2 + \|\widehat{A}\|_F^2 \\
&= \|\widehat{A}\|_F^2 - 2\text{Tr}(AP^* A^\top) + \text{Tr}(AP^{*2} A^\top) \\
&= \|\widehat{A}\|_F^2 - \sum_{i=1}^{n} (2\lambda_i^* - \lambda_i^{*2}) \langle A^T A, u_i u_i^T \rangle = \|\widehat{A}\|_F^2 - \sum_{i=1}^{n} \bar{\lambda} \langle A^T A, u_i u_i^T \rangle,
\end{aligned}$$

where the last inequality is by the choice of $\lambda_j^* = 1 - \sqrt{1 - \bar{\lambda}_j}$. The same equality holds true for group $B$. Therefore, $P^*$ gives the equal loss of $z^* \leq \hat{z}$ for two groups. The embedding $x \to (x \cdot u_1, \ldots, x \cdot u_{d-1}, \sqrt{\lambda_d^*}\, x \cdot u_d, \sqrt{\lambda_{d+1}^*}\, x \cdot u_{d+1})$ corresponds to the affine projection of any point (row) of $A, B$ defined by the solution $P^*$.

In both cases, the objective value is at most that of the original fairness objective. □

The result of Theorem 4.5.1 in two groups generalizes to more than two groups as follows. Given $m$ data points in $\mathbb{R}^n$ with $k$ subgroups $A_1, A_2, \ldots, A_k$, and $d \leq n$ the desired number of dimensions of projected space, we generalize Definition 4.4.4 of fair PCA problem as optimizing

$$\min_{U \in \mathbb{R}^{m \times n}, \text{rank}(U) \leq d} \max_{i \in \{1, \ldots, k\}} \left\{ \frac{1}{|A_i|} loss(A_i, U_{A_i}) \right\}, \tag{4.12}$$

where $U_{A_i}$ are matrices with rows corresponding to rows of $U$ for groups $A_i$.

**Theorem 4.5.2.** *There is a polynomial-time algorithm to find a projection such that it is of*

*dimension at most* $d+k-1$ *and achieves the optimal fairness objective value for dimension* $d$*.*

In contrast to the case of two groups, when there are more than two groups in the data, it is possible that all optimal solutions to fair PCA will not assign the same loss to all groups. However, with $k-1$ extra dimensions, we can ensure that the loss of each group remains at most the optimal fairness objective in $d$ dimension. The result of Theorem 4.5.2 follows by extending algorithm in Theorem 4.5.1 by adding linear constraints to SDP and LP for each extra group. An extreme solution $(\bar{\lambda}, z^*)$ of the resulting LP contains at most $k$ of $\lambda_i$'s that are strictly in between 0 and 1. Therefore, the final projection matrix $P^*$ has rank at most $d+k-1$.

**Runtime** We now analyze the runtime of Algorithm 8, which consists of solving SDP (4.4) and finding an extreme solution to an LP (4.5)-(4.11). The SDP and LP can be solved up to additive error of $\epsilon > 0$ in the objective value in $O(n^{6.5}\log(1/\epsilon))$ [26] and $O(n^{3.5}\log(1/\epsilon))$ [27] time, respectively. The running time of SDP dominates the algorithm both in theory and practice, and is too slow for practical uses for moderate size of $n$.

We propose another algorithm of solving SDP using the multiplicative weight (MW) update method. In theory, our MW takes $O(\frac{1}{\epsilon^2})$ iterations of solving standard PCA, giving a total of $O(\frac{n^3}{\epsilon^2})$ runtime, which may or may not be faster than $O(n^{6.5}\log(1/\epsilon))$ depending on $n, \epsilon$. In practice, however, we observe that after appropriately tuning one parameter in MW, the MW algorithm achieves accuracy $\epsilon < 10^{-5}$ within tens of iterations, and therefore is used to obtain experimental results in this chapter. Our MW can handle data of dimension up to a thousand with running time in less than a minute. Here, we discuss the details of implementation and analysis of the MW method.

## 4.6 Improved runtime of semi-definite relaxation by multiplicative weight update method

In this section, we show the multiplicative weight (MW) algorithm and runtime analysis to solve the fair PCA relaxation in two groups for $n \times n$ matrix up to $\epsilon$ additive error in $O(\frac{1}{\epsilon^2})$ iterations of solving a standard PCA, such as Singular Value Decomposition (SVD). Because SVD takes $O(n^3)$ time, the SDP relaxation (4.4) for two groups can be solved in $O(\frac{n^3}{\epsilon^2})$. Comparing to $O(n^{6.5}\log(1/\epsilon))$ runtime of an SDP solver that is commonly implemented with the interior point method [26], our algorithm may be faster or slower depending on $n, \epsilon$. In practice, however, we tune the parameter of MW algorithm much more aggressively than in theory, and often take the last iterate solution of MW rather the average when the last iterate performs better, which gives a much faster convergence rate. Our runs of MW show that MW converges in at most 10-20 iterations. Therefore, we use MW to implement our fair PCA algorithm. We note at the conclusion of this section that the algorithm and analysis can be extended to solving fair PCA in $k$ groups up to additive error $\epsilon$ in $O(\frac{\log k}{\epsilon^2})$ iterations.

Technically, the number of iterations for $k$ groups is $O(\frac{W^2 \log k}{\epsilon^2})$, where $W$ is the width of the problem, as defined in [87]. $W$ can usually be bounded by the maximum number of input or the optimal objective value. For our purpose, if the total variance of input data over all dimension is $L$, then the width $W$ is at most $L$. For simplicity, we assume $L \leq 1$ (e.g. by normalization in prepossessing step), hence obtaining the $O(\frac{\log k}{\epsilon^2})$ bound on number of iterations.

We first present an algorithmic framework and the corresponding analysis in the next two subsections, and later apply those results to our specific setting of solving the SDP (4.4) from fair PCA problem. The previous work by [87] shows how we may solve a feasibility problem of an LP using MW technique. Our main theoretical contribution is to propose and analyze the optimization counterpart of the feasibility problem, and the MW algorithm

we need to solve such problem. The MW we develop fits more seamlessly into our fair PCA setting and simplifies the algorithm to be implemented for solving the SDP (4.4).

### 4.6.1 Problem setup and oracle access

We first formulate the feasibility problem and its optimization counterpart in this section. The previous and new MW algorithms and their analysis are presented in the following Section 4.6.3.

**Previous work: multiplicative weight on feasibility problem**

**Problem.** As in [87], we are given $A \in \mathbb{R}^{m\times n}$ as an $m \times n$ real matrix, $x \in \mathbb{R}^n, b \in \mathbb{R}^m$, and $\mathcal{P}$ as a convex set in $\mathbb{R}^n$, and the goal is to check the feasibility problem

$$\exists ? x \in \mathcal{P} \ : \ Ax \geq b \tag{4.13}$$

by giving a feasible $x \in \mathcal{P}$ or correctly deciding that such $x$ does not exist.

**Oracle Access.** We assume the existence if an oracle that, given any probability vector $p \in \Delta_m$ over $m$ constraints of (4.13), correctly answers a single-constraint problem

$$\exists ? x \in \mathcal{P} \ : \ p^\top Ax \geq p^\top b \tag{4.14}$$

by giving a feasible $x \in \mathcal{P}$ or correctly deciding that such $x$ does not exist. We may think of (4.14) as a weighted version of (4.13), with weights on each constraint $i \in [m]$ being $p_i$.

As (4.14) consists of only one constraint, solving (4.14) is much easier than (4.13) in many settings. For example, in our PCA setting, solving (4.4) directly is non-trivial, but the weighted version (4.14) is a standard PCA problem: we weight each group $A, B$ based on $p$, and then apply a PCA algorithm (Singular Value Decomposition) on the sum of two

weighted groups. The solution gives an optimal value of $p^\top Ax - p^\top\ b$ in (4.14). More details of application in fair PCA settings are in Section 4.6.4

### 4.6.2 New setting: multiplicative weight on optimization problem

**Problem.** The previous work gives an MW framework for the feasibility question. Here we propose an optimization framework, which asks for the best $x \in \mathcal{P}$ rather than an existence of $x \in \mathcal{P}$. The optimization framework can be formally stated as, given $A \in \mathbb{R}^{m\times n}$ as an $m \times n$ real matrix, $x \in \mathbb{R}^n, b \in \mathbb{R}^m$, and $\mathcal{P}$ as a convex set in $\mathbb{R}^n$, we need to solve

$$\min z \ : \ Ax - b + z \cdot \mathbf{1} \geq 0, \text{ s.t. } x \in \mathcal{P} \tag{4.15}$$

where $\mathbf{1}$ denotes the $m \times 1$ vector with entries 1. Denote $z^*$ the optimum of (4.15).

With the same type of oracle access, we may run (4.13) for $O(\log \frac{n}{\epsilon})$ iterations to do binary search for the correct value of optimum $z^*$ up to an additive error $\epsilon$. However, our main contribution is to modify the previous multiplicative weight algorithm and the definition of the oracle to solve (4.15) without guessing the optimum $z^*$. This improves the runtime slightly (reduce the $\log(n/\epsilon)$ factor) and simplifies the algorithm.

**Feasibility Oracle Access.** We assume the existence of an oracle that, given any probability vector $p \in \Delta_m$ over $m$ constraints of (4.15), correctly answers a single-constraint problem

$$\text{Find } x \in \mathcal{P} \ : \ p^\top Ax - p^\top b + z^* \geq 0 \tag{4.16}$$

There is always such $x$ because multiplying (4.15) on the left by $p^\top$ shows that one of such $x$ is the optimum $x^*$ of (4.15). However, finding one may not be as trivial as asserting problem's feasibility. In general, (4.16) can be tricky to solve since we do not yet know the value of $z^*$.

**Optimization Oracle Access.** We define the oracle that, given $p \in \Delta_m$ over $m$ constraints of (4.15), correctly answers one maximizer of

$$\min z \ : \ p^\top Ax - p^\top b + z \geq 0, \text{ s.t. } x \in \mathcal{P} \tag{4.17}$$

which is stronger than and is sufficient to solve (4.16). This is because $x^*$ of (4.15) is one feasible $x$ to (4.17), so the optimum $\hat{z}$ of (4.17) is at most $z^*$. Therefore, the optimum $x$ by (4.17) can be a feasible solution to (4.16). In many setting, because (4.16) is only one-constraint problem, it is possible to solve the optimization version (4.17) instead. For example, in our fair PCA on two groups setting, we can solve the (4.17) by standard PCA on the union of two groups after an appropriate weighting on each group. More details of application in fair PCA settings are in Section 4.6.4.

### 4.6.3 Algorithm and Analysis

The line of proof follows similarly from [87]. We first state the technical property that the oracle satisfies in our optimization framework, then show how to use that property to bound the number of iterations. We fix $A \in \mathbb{R}^{m \times n}$ as an $m \times n$ real matrix, $x \in \mathbb{R}^n, b \in \mathbb{R}^m$, and $\mathcal{P}$ is a convex set in $\mathbb{R}^n$

**Definition 4.6.1.** *(analogous to [87]) An $(\ell, \rho)$-bounded* **oracle** *for parameter $0 \leq \ell \leq \rho$ is an algorithm which, given $p \in \Delta_m$, solve* (4.16)*. Also, there is a fixed $I \subseteq [m]$ (i.e. fixed across all possible $p \in \Delta_m$) of constraints such that for all $x \in \mathcal{P}$ output by this algorithm,*

$$\forall i \in I : \ A_i x - b_i + z^* \in [-\ell, \rho] \tag{4.18}$$

$$\forall i \notin I : \ A_i x - b_i + z^* \in [-\rho, \ell] \tag{4.19}$$

Note that even though we do not know $z^*$, if we know the range of $A_i x - b_i$ for all $i$, we can bound the range of $z^*$. Therefore, we can still find a useful $\ell, \rho$ that an oracle satisfies.

Now we are ready to state the main result of this section: that we may solve the optimization version by multiplicative update as quickly as solving the feasibility version of the problem.

**Theorem 4.6.2.** *Let $\epsilon > 0$ be given. Suppose there exists $(\ell, \rho)$-bounded* **oracle** *and $\ell \geq \epsilon/4$ to solving* (4.16)*. Then there exists an algorithm that solves* (4.15) *up to additive error $\epsilon$, i.e. outputs $x \in \mathcal{P}$ such that*

$$Ax - b + z^* \cdot \mathbf{1} \geq -\epsilon \tag{4.20}$$

*The algorithm calls* **oracle** $O(\ell\rho \log(m)/\epsilon^2)$ *times and has additional $O(m)$ time per call.*

*Proof.* The proof follows similarly as Theorem 3.3 in [87], but we include details here for completeness. The algorithm is multiplicative update in nature, as in equation (2.1) of [87]. The algorithm starts with uniform $p^0 \in \Delta_m$ over $m$ constraints. Each step the algorithm asks the **oracle** with input $p^t$ and receive $x^t \in \mathcal{P}$. We use the loss vector $m^t = \frac{1}{\rho}(Ax^t - b)$ to update the weight $p^t$ for the next step with learning rate $\eta$. After $T$ iterations (which will be specified later), the algorithm outputs $\bar{x} = \frac{1}{T}\sum_{t=1}^{T} x^t$.

Note that using either the loss $\frac{1}{\rho}(Ax^t - b + z^*)$ and $\frac{1}{\rho}(Ax^t - b)$ behaves the same algorithmically due to the renormalization step on the vector $(p_i^t)_{i=1}^m$. Therefore, just for analysis, we use a hypothetical loss $m^t = \frac{1}{\rho}(Ax^t - b + z^*)$ to update $p^t$ (this loss can't be used algorithmically since we do not know $z^*$). By Theorem 2.1 in [87], for each constraint $i \in [m]$ and all $\eta \leq 1/2$,

$$\begin{aligned}\sum_{t=1}^{T} m^t \cdot p^t &\leq \sum_{t=1}^{T} m_i^t + \eta \sum_{t=1}^{T} |m_i^t| + \frac{\log m}{\eta} \\ &= \frac{1}{\rho}\sum_{t=1}^{T}(A_i x^t - b_i + z^*) + \frac{\eta}{\rho}\sum_{t=1}^{T}|A_i x^t - b_i + z^*| + \frac{\log m}{\eta}\end{aligned} \tag{4.21}$$

By property (4.16) of the **oracle**,

$$\sum_{t=1}^{T} m^t \cdot p^t = \frac{1}{\rho} \sum_{t=1}^{T} \left( (p^t)^\top (Ax^t - b) + z^* \right) \geq 0 \tag{4.22}$$

We now split into two cases. If $i \in I$, then (4.21) and (4.22) imply

$$\begin{aligned} 0 &\leq \frac{1+\eta}{\rho} \sum_{t=1}^{T} (A_i x^t - b_i + z^*) + \frac{2\eta}{\rho} \sum_{t: A_i x^t - b_i < 0} |A_i x^t - b_i + z^*| + \frac{\log m}{\eta} \\ &\leq \frac{1+\eta}{\rho} T(A_i \bar{x} - b_i + z^*) + \frac{2\eta}{\rho} T\ell + \frac{\log n}{\eta} \end{aligned}$$

Multiplying the last inequality by $\frac{\rho}{T}$ and rearranging terms, we have

$$0 \leq (1+\eta)(A_i \bar{x} - b_i + z^*) + 2\eta\ell + \frac{\rho \log m}{T\eta} \tag{4.23}$$

If $i \notin I$, then (4.21) and (4.22) imply

$$\begin{aligned} 0 &\leq \frac{1-\eta}{\rho} \sum_{t=1}^{T} (A_i x^t - b_i + z^*) + \frac{2\eta}{\rho} \sum_{t: A_i x^t - b_i > 0} |A_i x^t - b_i + z^*| + \frac{\log m}{\eta} \\ &\leq \frac{1-\eta}{\rho} T(A_i \bar{x} - b_i) + \frac{2\eta}{\rho} T\ell + \frac{\log n}{\eta} \end{aligned}$$

Multiplying inequality by $\frac{\rho}{T}$ and rearranging terms, we have

$$0 \leq (1-\eta)(A_i \bar{x} - b_i + z^*) + 2\eta\ell + \frac{\rho \log m}{T\eta} \tag{4.24}$$

To use (4.23) and (4.24) to show that $A_i \bar{x} - b_i + z^*$ is close to 0 simultaneously for two cases, pick $\eta = \frac{\epsilon}{8\ell}$ (note that $\eta \leq 1/2$ by requiring $\ell \geq \epsilon/4$, so we may apply Theorem 2.1

in [87]). Then for all $T \geq \frac{4\rho \log(m)}{\epsilon\eta} = \frac{32\ell\rho \log(m)}{\epsilon^2}$, we have

$$2\eta\ell + \frac{\rho \log m}{T\eta} \leq \frac{\epsilon}{4} + \frac{\epsilon}{4} = \frac{\epsilon}{2} \tag{4.25}$$

Hence, (4.23) implies

$$0 \leq (1+\eta)(A_i\bar{x} - b_i + z^*) + \frac{\epsilon}{2} \Rightarrow A_i\bar{x} - b_i + z^* \geq -\frac{\epsilon}{2} \tag{4.26}$$

and (4.24) implies

$$0 \leq (1-\eta)(A_i\bar{x} - b_i + z^*) + \frac{\epsilon}{2} \Rightarrow A_i\bar{x} - b_i + z^* \geq -\epsilon \tag{4.27}$$

using the fact that $\eta \leq 1/2$. □

### 4.6.4 Application of multiplicative update method to the fair PCA problem

In this section, we apply MW results for solving LP to solve the SDP relaxation (4.4) of fair PCA.

**LP formulation of fair PCA relaxation.** The SDP relaxation (4.4) of fair PCA can be written in the form (4.15) as an LP with two constraints

$$\min_{P \in \mathcal{P}, z \in \mathbb{R}} z \text{ s.t.} \tag{4.28}$$

$$z \geq \alpha - \frac{1}{m_1}\langle A^\top A, P\rangle \tag{4.29}$$

$$z \geq \beta - \frac{1}{m_2}\langle B^\top B, P\rangle \tag{4.30}$$

for some constants $\alpha, \beta$, where the feasible region of variables is over a set of PSD matrices:

$$\mathcal{P} = \{M \in \mathbb{R}^{n\times n} : 0 \preceq M \preceq I, \text{tr}(M) \leq d\} \tag{4.31}$$

We will apply the multiplicative weight algorithm to solve (4.28)-(4.30).

**Oracle Access.** First, we present an the oracle in Algorithm 9, which is in the form (4.17) and therefore can be used to solve (4.16). As defined in (4.17), the optimization oracle, given a weight vector $p = (p_1, p_2) \in \Delta_2$, should be able to solve the LP with one weighted constraint obtained from weighting two constraints (4.29) and (4.30) by $p$. However, because both constraints involve only dot products of same variable $P$ with constant matrices $A^\top A$ and $B^\top B$, which are linear functions, the weighted constraint will involve the dot product of the same variable $P$ with weighted sum of those constant matrices $\frac{p_1}{m_1} A^\top A + \frac{p_2}{m_2} B^\top B$.

**Algorithm 9:** Fair PCA oracle (oracle to Algorithm 10)

**Input :** $p = (p_1, p_2) \in \Delta_2, \alpha, \beta \in \mathbb{R},\ A \in \mathbb{R}^{m_1 \times n}, B \in \mathbb{R}^{m_2 \times n}$

**Output:** $\arg\min_{P, z_1, z_2}\ p_1 z_1 + p_2 z_2,$ subject to

$z_1 = \alpha - \frac{1}{m_1}\langle A^\top A, P\rangle,$

$z_2 = \beta - \frac{1}{m_2}\langle B^\top B, P\rangle,$

$P \in \mathcal{P} = \{M \in \mathbb{R}^{n \times n} : 0 \preceq M \preceq I, \mathrm{Tr}(M) \leq d\}$

Set $V \in \mathbb{R}^{n \times d}$ to be the matrix with top $d$ principles components of $\frac{p_1}{m_1} A^\top A + \frac{p_2}{m_2} B^\top B$ as columns;

**return** $P^* = VV^\top$ , $z_1^* = \alpha - \frac{1}{m_1}\langle A^\top A, P^*\rangle, z_2^* = \beta - \frac{1}{m_2}\langle B^\top B, P^*\rangle$;

**MW Algorithm.** Our multiplicative weight update algorithm for solving fair PCA relaxation (4.28)-(4.30) is presented in Algorithm 10. The algorithm follows exactly from the construction in Theorem 4.6.2. The runtime analysis of our MW Algorithm 10 follows directly from the same theorem.

**Corollary 4.6.3.** *Let $\epsilon > 0$. Algorithm 10 finds a near-optimal (up to additive error of $\epsilon$) solution $P$ to* (4.28)-(4.30) *in $O\left(\frac{1}{\epsilon^2}\right)$ iterations of solving standard PCA, and therefore in $O(\frac{n^3}{\epsilon^2})$ running time.*

*Proof.* We first check that the oracle presented in Algorithm 9 satisfies $(\ell, \rho)$-boundedness and find those parameters. We may normalize the data so that the variances of $\frac{A^\top A}{m_1}$ and $\frac{B^\top B}{m_2}$

**Algorithm 10:** Multiplicative weight update for fair PCA

**Input :** $\alpha, \beta \in \mathbb{R},\ A \in \mathbb{R}^{m_1 \times n}, B \in \mathbb{R}^{m_2 \times n}, \eta > 0$, positive integer $T$

**Output:** $\arg\min_{P,z}\ z,$ subject to

$z \geq \alpha - \frac{1}{m_1}\langle A^\top A, P\rangle,$

$z \geq \beta - \frac{1}{m_2}\langle B^\top B, P\rangle,$

$P \in \mathcal{P} = \{M \in \mathbb{R}^{n\times n} : 0 \preceq M \preceq I, \mathrm{Tr}(M) \leq d\}$

Initialize $p^0 = (1/2, 1/2)$;

**for** $t = 1, \ldots, T$ **do**

  $(P_t, m_1^t, m_2^t) \leftarrow \mathbf{oracle}(p^{t-1}, \alpha, \beta, A, B)$;

  $\hat{p}_i^t \leftarrow p_i^{t-1} e^{\eta m_i^t}$, for $i = 1, 2$;

  $p_i^t \leftarrow \hat{p}_i^t / (\hat{p}_1^t + \hat{p}_2^t)$, for $i = 1, 2$;

**end**

**return** $P^* = \frac{1}{T}\sum_{t=1}^{T} P_t$ , $z^* = \max\{\alpha - \frac{1}{m_1}\langle A^\top A, P^*\rangle, \beta - \frac{1}{m_2}\langle B^\top B, P^*\rangle\}$

are bounded by 1. Therefore, for any PSD matrix $P \preceq I$, we have $\frac{1}{m_1}\langle A^\top A, P\rangle \leq 1$. In addition, in the application to fair PCA setting, we have $\alpha = \frac{\|\widehat{A}\|_F^2}{m_1}$. Hence, $\frac{1}{m_1}\langle A^\top A, P\rangle \leq \alpha$ for any feasible $P \in \mathcal{P} = \{M \in \mathbb{R}^{n\times n} : 0 \preceq M \preceq I, \mathrm{Tr}(M) \leq d\}$ by the definition of $\widehat{A}$ (recall Definition 4.3.1). Therefore,

$$0 \leq \alpha - \frac{1}{m_1}\langle A^\top A, P\rangle \leq 1, \forall P \in \mathcal{P} \tag{4.32}$$

and similarly $\beta - \frac{1}{m_2}\langle B^\top B, P\rangle \in [0, 1]$. Hence, the optimal solution of Algorithm 10 satisfies $z^* \in [0, 1]$. Therefore, the oracle is $(1, 1)$-bounded.

Next we analyze the runtime of Algorithm 10. By Theorem 4.6.2, Algorithm 10 calls the oracle $O(1/\epsilon^2)$ times. The bottleneck in an oracle call is solving PCA on the weighted sum of two groups, which takes $O(n^3)$ time. The additional processing time to update the weight is negligible compared to this $O(n^3)$ time for solving PCA. □

**MW for More Than Two Groups.** Algorithms 9 and 10 can be naturally extended to $k$ groups. Theorem 4.6.2 states that we need $O(\frac{\log k}{\epsilon^2})$ calls to the oracle with additional $O(k)$ time per call (to update the weight for each loss). In each call, we must compute the weighted sum of $k$ matrices of dimension $n \times n$, which takes $O(kn^2)$ arithmetic operations

and perform SVD. In natural settings, $k$ is much smaller than $n$, and hence the runtime $O(n^3)$ of SVD in each oracle call will dominate.

## 4.7 Experiments

We use two common human-centric data sets for our experiments. The first one is labeled faces in the wild (LFW) [88], the second is the Default Credit data set [89]. We preprocess all data to have its mean at the origin. For the LFW data, we normalized each pixel value by $\frac{1}{255}$. The gender information for LFW was taken from [90], who manually verified the correctness of these labels. For the credit data, since different attributes are measurements of incomparable units, we normalized the variance of each attribute to be equal to 1.

**Results** We focus on projections into relatively few dimensions, as those are used ubiquitously in early phases of data exploration. As we already saw in Figure 4.1 left, at lower dimensions, there is a noticeable gap between PCA's average reconstruction error for men and women on the LFW data set. This gap is at the scale of up to 10% of the total reconstruction error when we project to 20 dimensions. This still holds when we subsample male and female faces with equal probability from the data set, and so men and women have equal magnitude in the objective function of PCA (Figure 4.1 right).

Figure 4.4 shows the average reconstruction error of each population (Male/Female, Higher/Lower education) as the result of running vanilla PCA and Fair PCA on LFW and Credit data. As we expect, as the number of dimensions increase, the average reconstruction error of every population decreases. For LFW, the original data is in 1764 dimensions ($49\times36$ images), therefore, at 20 dimensions we still see a considerable reconstruction error. For the Credit data, we see that at 21 dimensions, the average reconstruction error of both populations reach 0, as this data originally lies in 21 dimensions. In order to see how fair are each of these methods, we need to zoom in further and look at the average loss of populations.

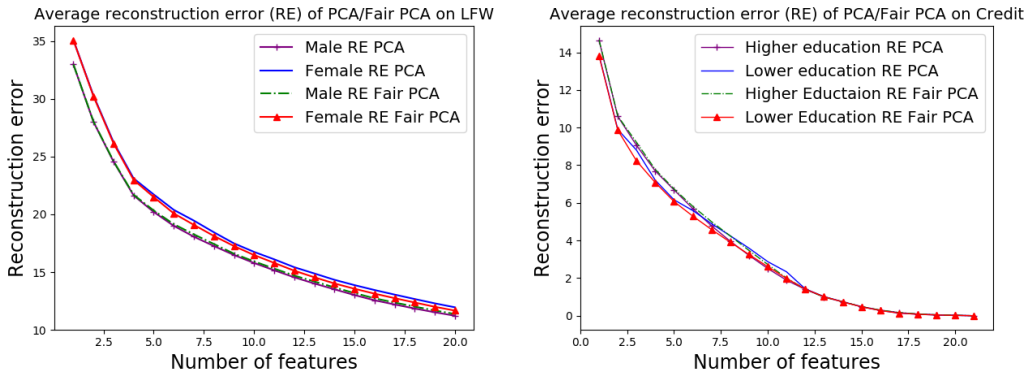



Figure 4.4: Reconstruction error of PCA/Fair PCA on LFW and the Default Credit data set.

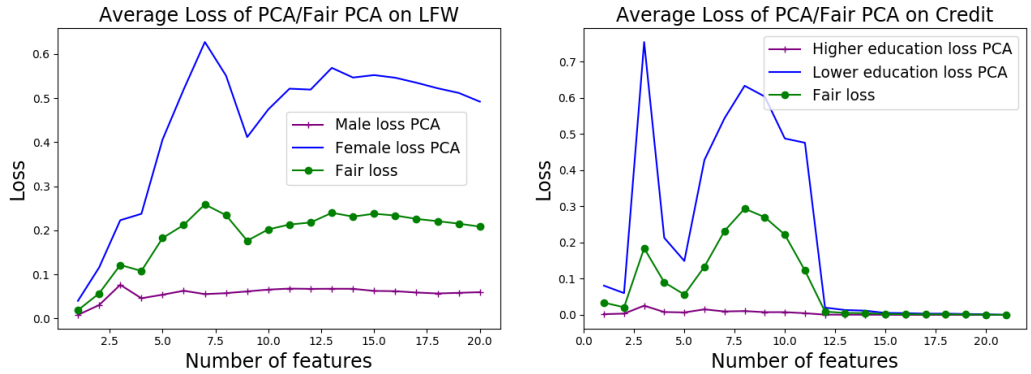



Figure 4.5: Loss of PCA/Fair PCA on LFW and the Default Credit data set.

Figure 4.5 shows the average loss of each population as the result of applying vanilla PCA and Fair PCA on both data sets. Note that at the optimal solution of Fair PCA, the average loss of two populations are the same, therefore we have one line for "Fair loss". We observe that PCA suffers much higher average loss for female faces than male faces. After running fair PCA, we observe that the average loss for fair PCA is relatively in the middle of the average loss for male and female. So, there is improvement in terms of the female average loss which comes with a cost in terms of male average loss. Similar observation holds for the Credit data set. In this context, it appears there is some cost to optimizing for the less well represented population in terms of the better-represented population.

## 4.8 Generalization and Improvements

This chapter is far from a complete study of when and how dimensionality reduction might help or hurt the fair treatment of different populations. Several concrete theoretical questions remain using our framework. What is the complexity of optimizing the fairness objective? Is it NP-hard, even for $d = 1$? Our work naturally extends to $k$ predefined subgroups rather than just 2, where the number of additional dimensions our algorithm uses is $k - 1$. Are these additional dimensions necessary for computational efficiency? On the more practical side, we would like to study how the representational bias of an early preprocessing data analysis technique such as PCA, would affect the overall bias of a machine learning pipeline. Such an observation would also allow us to measure the effectiveness of FAIR-PCA in ameliorating the total bias of the system.

In a follow-up work [91], we study the aforementioned open theoretical questions in the more general context of *multi-criteria dimensionality reduction*. In this section, we briefly discuss the more general setup and our advancements to answer the aforementioned open questions. We define the multi-criteria dimensionality reduction problem in the context where we are given multiple objectives that need to be optimized simultaneously. Direct applications of our model are Fair-PCA problem and the Nash Social Welfare (NSW) problem [92, 93] where in the latter the goal is to maximize the product of the individual variances of the groups achieved by the common low-dimensinal space.

In order to balance multiple objectives, we aim to find a projection $Q$[3]. Let $\mathcal{Q}_d$ denote the set of all $n \times d$ projection matrices $Q$, i.e., matrices with $d$ orthonormal columns. For each group $A_i$, we associate a function $f_i : \mathcal{Q}_d \to \mathbb{R}$ that denotes the group's objective value for a particular projection — this could simply be the variance of the group represention after projection as in FAIR-PCA . For any $g : \mathbb{R}^k \to \mathbb{R}$, we define the $(f, g)$-multi-criteria

[3]We are switching the notation here from the projection matrix $P_{n\times n}$ (used in previous sections of this chapter) to the projection matrix $Q_{n\times d}$

dimensionality reduction problem as finding a $d$-dimensional projection $Q$ that optimizes

$$\max_{Q \in \mathcal{Q}_d} g(f_1(Q), f_2(Q), \ldots, f_k(Q)). \qquad \text{(MULTI-CRITERIA-DIMENSION-REDUCTION)}$$

In the FAIR-PCA , $g$ is the $\min$ function and $f_i(Q) = \|A_i Q\|^2$ is the total squared norm of the projection of vectors in $A_i$.

Our first main result is regarding MULTI-CRITERIA-DIMENSION-REDUCTION when $g$ is monotone nondecreasing in any one coordinate and concave, and each $f_i$ is an affine function of $QQ^T$ (and thus a special case of a quadratic function in $Q$).

**Theorem 4.8.1.** *There is a polynomial time algorithm for* 2*-group* MULTI-CRITERIA-DIMENSION-REDUCTION *problem when* $g$ *is concave and monotone nondecreasing for at least one of its two arguments, and each* $f_i$ *is linear in* $QQ^T$*, i.e.,* $f_i(Q) = \langle B_i, QQ^T \rangle$ *for some matrix* $B_i(A)$.

For $k > 2$, the SDP might not recover a rank $d$ solution. Indeed, we show that the SDP may be inexact even for $k = 3$ . For $k > 2$, we can bound the rank of a solution to the SDP by $d + \left\lfloor \sqrt{2k + \frac{1}{4}} - \frac{3}{2} \right\rfloor$.

Finally, we show that MULTI-CRITERIA-DIMENSION-REDUCTION and even FAIR-PCA is NP-hard to solve exactly even for $d = 1$, when the number of groups is part of the input. For more detailed discussion of the MULTI-CRITERIA-DIMENSION-REDUCTION and the proofs of theorical results please refer to [91].

# CHAPTER 5
# GUARANTEES FOR SPECTRAL CLUSTERING WITH FAIRNESS CONSTRAINTS

Given the widespread popularity of spectral clustering (SC) for partitioning graph data, we study a version of constrained SC in which we try to incorporate the fairness notion proposed by [28]. According to this notion, a clustering is fair if every demographic group is approximately proportionally represented in each cluster. To this end, we develop variants of both normalized and unnormalized constrained SC and show that they help find fairer clusterings on both synthetic and real data. We also provide a rigorous theoretical analysis of our algorithms. While there have been efforts to incorporate various constraints into the SC framework, theoretically analyzing them is a challenging problem. We overcome this by proposing a natural variant of the stochastic block model where $h$ groups have strong inter-group connectivity, but also exhibit a "natural" clustering structure which is fair. We prove that our algorithms can recover this fair clustering with high probability.

## 5.1 Introduction

Machine learning (ML) has recently seen an explosion of applications in settings to guide or make choices directly affecting people. Examples include applications in lending, marketing, education, and many more. Close on the heels of the adoption of ML methods in these everyday domains have been any number of examples of ML methods displaying unsavory behavior towards certain demographic groups. These have spurred the study of *fairness* of machine learning algorithms. Numerous mathematical formulations of fairness have been proposed for supervised learning settings, each with their strengths and shortcomings in terms of what they disallow and how difficult they may be to satisfy (e.g., [76, 78, 94, 79]. Somewhat more recently, the community has begun to study appropriate notions of fairness

for unsupervised learning settings (e.g., [28, 95, 96]; see Section 5.5).

In particular, the recent work of [28] proposes a notion of fairness for clustering requiring each cluster to have proportional representation from different demographic groups. Their paper provides approximation algorithms for $k$-center and $k$-median clustering that incorporate this fairness notion. The follow-up work of [30] extends this to $k$-means clustering. These papers open up an important line of work that aims at studying the following questions for clustering: a) *How to incorporate fairness constraints into popular clustering objectives and algorithms?* and b) *What is the price of fairness?* For example, the results of [28] indicate that achieving fair clusterings comes at a significant increase in the $k$-center/$k$-median objective value. While the above works focus on clustering data sets in Euclidean/metric spaces, a large body of clustering problems involve graph data. On such data, spectral clustering [SC; 97] is the method of choice in practice. In this chapter, we extend the above line of work by studying the implications of incorporating the fairness notion of [28] into SC.

The contributions of this chapter are as follows:

- We show how to incorporate the constraints that in each cluster, every group should be represented with the same proportion as in the original data set into the SC framework. For continuity with prior work (as discussed above; also see Section 5.5), we refer to these constraints as *fairness* constraints and speak of *fair* clusterings. However, the terms *proportionality* and *proportional* would be a more formal description of our goal.

  Our approach to incorporate the fairness constraints is analogous to existing versions of constrained SC that try to incorporate must-link constraints (see Section 5.5). In contrast to the work of [28], which always yields a fair clustering no matter how much the objective value increases compared to an unfair clustering, our approach does not guarantee that we end up with a fair clustering. Rather, our approach guides SC to find a good *and* fair clustering if such a one exists.

- Indeed, we prove that our algorithms find a good and fair clustering in a natural variant of the famous stochastic block model that we propose. In our variant, $h$ demographic groups have strong inter-group connectivity, but also exhibit a "natural" clustering structure that is fair. We provide a rigorous analysis of our algorithms showing that they can recover this fair clustering with high probability. To the best of our knowledge, such an analysis has not been done before for constrained versions of SC.

- We conclude by giving experimental results on real-world data sets where proportional clustering can be a desirable goal, comparing the proportionality and objective value of standard SC to our methods. Our experiments confirm that our algorithms tend to find fairer clusterings compared to standard SC. A surprising finding is that in many real data sets *achieving higher proportionality often comes at minimal cost*, namely, that our methods produce clusterings that are fairer, but have objective values very close to those of clusterings produced by standard SC. This complements the results of [28], where achieving fairness constraints *exactly* comes at a significant cost in the objective value, and indicates that in some scenarios fairness and objective value need not be at odds with one other.

**Notation** For $n \in \mathbb{N}$, we use $[n] = \{1, \ldots, n\}$. $I_n$ denotes the $n \times n$-identity matrix and $0_{n\times m}$ is the $n \times m$-zero matrix. $\mathbf{1}_n$ denotes a vector of length $n$ with all entries equaling 1. For a matrix $A \in \mathbb{R}^{n\times m}$, we denote the transpose of $A$ by $A^T \in \mathbb{R}^{m\times n}$. For $A \in \mathbb{R}^{n\times n}$, $\mathrm{tr}(A)$ denotes the trace of $A$, that is $\mathrm{tr}(A) = \sum_{i=1}^{n} A_{ii}$. If we say that a matrix is positive (semi-)definite, this implies the matrix is symmetric.

## 5.2 Spectral clustering

To set the ground and introduce terminology, we review Spectral Clustering (SC). There are several versions of SC [97]. For ease of presentation, here we focus on unnormalized

SC [98]. In section 5.7, we adapt all findings of this section and the following Section 5.3 to normalized SC [99].

Let $G = (V, E)$ be an undirected graph on $V = [n]$. We assume that each edge between two vertices $i$ and $j$ carries a positive weight $W_{ij} > 0$ encoding the strength of similarity between the verices. If there is no edge between $i$ and $j$, we set $W_{ij} = 0$. We assume that $W_{ii} = 0$, for all $i \in [n]$. Given $k \in \mathbb{N}$, unnormalized SC aims to partition $V$ into $k$ clusters with minimum value of the RatioCut objective function as follows [see 97, for details]: for a clustering $V = C_1 \dot{\cup} \ldots \dot{\cup} C_k$ we have

$$\mathrm{RatioCut}(C_1, \ldots, C_k) = \sum_{l=1}^{k} \frac{\mathrm{Cut}(C_l, V \setminus C_l)}{|C_l|}, \tag{5.1}$$

where $\mathrm{Cut}(C_l, V \setminus C_l)$ denotes the weight of the cut $(C_l, V \setminus C_l)$, that is

$$\mathrm{Cut}(C_l, V \setminus C_l) = \sum_{i \in C_l, j \in V \setminus C_l} W_{ij}.$$

Let $W = (W_{ij})_{i,j \in [n]}$ be the weighted adjacency matrix of $G$ and $D$ be the degree matrix, that is a diagonal matrix with the vertex degrees $d_i = \sum_{j \in [n]} W_{ij}$, $i \in [n]$, on the diagonal. Let $L = D - W$ denote the unnormalized graph Laplacian matrix. Note that $L$ is positive semi-definite. A key insight is that if we encode a clustering $V = C_1 \dot{\cup} \ldots \dot{\cup} C_k$ by a matrix $H \in \mathbb{R}^{n \times k}$ with

$$H_{il} = \begin{cases} 1/\sqrt{|C_l|}, & i \in C_l, \\ 0, & i \notin C_l \end{cases}, \tag{5.2}$$

then $\mathrm{RatioCut}(C_1, \ldots, C_k) = \mathrm{Tr}(H^T L H)$. Hence, in order to minimize the RatioCut function over all possible clusterings, we could instead solve

$$\min_{H \in \mathbb{R}^{n \times k}} \mathrm{Tr}(H^T L H) \text{ subject to } H \text{ is of form (5.2).} \tag{5.3}$$

**Algorithm 11:** Unnormalized SC

**Input**:weighted adjacency matrix $W \in \mathcal{R}^{n \times n}$ $k \in \mathcal{N}$
**Output**:a clustering of $[n]$ into $k$ clusters

- Compute the Laplacian matrix $L = D - W$
- Compute the $k$ smallest (respecting multiplicities) eigenvalues of $L$ and the corresponding orthonormal eigenvectors (written as columns of $H \in \mathbb{R}^{n \times k}$)
- Apply $k$-means clustering to the rows of $H$

Spectral clustering relaxes this minimization problem by replacing the requirement that $H$ has to be of form (5.2) with the weaker requirement that $H^T H = I_k$, that is it solves

$$\min_{H \in \mathbb{R}^{n \times k}} \mathrm{Tr}(H^T L H) \text{ subject to } H^T H = I_k. \tag{5.4}$$

Since $L$ is symmetric, it is well known that a solution to (5.4) is given by the matrix $H$ that contains some orthonormal eigenvectors corresponding to the $k$ smallest eigenvalues (respecting multiplicities) of $L$ as columns [100, Section 5.2.2]. Consequently, the first step of SC is to compute such an optimal $H$ by computing the $k$ smallest eigenvalues and corresponding eigenvectors. The second step is to infer a clustering from $H$. While there is a one-to-one correspondence between a clustering and a matrix of the form (5.2), this is not the case for a solution $H$ to the relaxed problem (5.4). Usually, a clustering of $V$ is inferred from $H$ by applying $k$-means clustering to the rows of $H$. We summarize unnormalized SC as Algorithm 11. Note that, in general, there is no guarantee on how close the RatioCut value of the clustering obtained by Algorithm 11 to the RatioCut value of an optimal clustering (solving (5.3)) is.

### 5.3 Adding fairness constraints

We now extend the above setting to incorporate fairness constraints. Suppose that the data set $V$ contains $h$ groups $V_s$ such that $V = \dot{\cup}_{s \in [h]} V_s$. [28] proposed a notion of fairness for clustering asking that every cluster contains approximately the same number of elements

from each group $V_s$. For a clustering $V = C_1 \dot\cup \ldots \dot\cup C_k$, define the balance of cluster $C_l$ as

$$\text{balance}(C_l) = \min_{s \neq s' \in [h]} \frac{|V_s \cap C_l|}{|V_{s'} \cap C_l|} \in [0, 1]. \tag{5.5}$$

The higher the balance of each cluster, the fairer is the clustering according to the notion of [28]. For any clustering, we have $\min_{l \in [k]} \text{balance}(C_l) \leq \min_{s \neq s' \in [h]} |V_s|/|V_{s'}|$, so that this fairness notion is actually asking for a clustering in which in every cluster, each group is (approximately) represented with the same fraction as in the whole data set $V$. The following lemma shows how to incorporate this goal into the RatioCut minimization problem (5.3) using a linear constraint on $H$.

**Lemma 5.3.1** (Fairness constraints as linear constraint on $H$)**.** *For $s \in [h]$, let $f^{(s)} \in \{0, 1\}^n$ be the group-membership vector of $V_s$, that is $f_i^{(s)} = 1$ if $i \in V_s$ and $f_i^{(s)} = 0$ otherwise. Let $V = C_1 \dot\cup \ldots \dot\cup C_k$ be a clustering that is encoded as in* (5.2)*. We have, for every $l \in [k]$,*

$$\forall s \in [h-1] : \sum_{i=1}^{n} \left( f_i^{(s)} - \frac{|V_s|}{n} \right) H_{il} = 0 \quad \Leftrightarrow$$
$$\forall s \in [h] : \frac{|V_s \cap C_l|}{|C_l|} = \frac{|V_s|}{n}.$$

*Proof.* This simply follows from

$$\sum_{i=1}^{n} \left( f_i^{(s)} - \frac{|V_s|}{n} \right) H_{il} = \frac{|V_s \cap C_l|}{\sqrt{|C_l|}} - \frac{|V_s| \cdot |C_l|}{n\sqrt{|C_l|}}$$

and $|C_l| = \sum_{s=1}^{h} |V_s \cap C_l|$. □

Consequently, if we want to find a clustering that minimizes the RatioCut objective

**Algorithm 12:** Unnormalized SC with fairness constraints

**Input:** weighted adjacency matrix $W \in \mathbb{R}^{n \times n}$; $k \in \mathbb{N}$; group-membership vectors $f^{(s)} \in \{0,1\}^n$, $s \in [h]$

**Output:** a clustering of $[n]$ into $k$ clusters

- Compute the Laplacian matrix $L = D - W$
- Let $F$ be a matrix with columns $f^{(s)} - \frac{|V_s|}{n} \cdot \mathbf{1}_n$, $s \in [h-1]$
- Compute a matrix $Z$ whose columns form an orthonormal basis of the nullspace of $F^T$
- Compute the $k$ smallest (respecting multiplicities) eigenvalues of $Z^T L Z$ and the corresponding orthonormal eigenvectors (written as columns of $Y$)
- Apply $k$-means clustering to the rows of $H = ZY$

function and is as fair as possible, we have to solve

$$\min_{H \in \mathbb{R}^{n \times k}} \operatorname{Tr}(H^T L H) \text{ subject to } H \text{ is of form (5.2)} \text{ and } F^T H = 0_{(h-1)\times k}, \tag{5.6}$$

where $F \in \mathbb{R}^{n \times (h-1)}$ is the matrix that has the vectors $f^{(s)} - (|V_s|/n) \cdot \mathbf{1}_n$, $s \in [h-1]$, as columns. In the same way as we have relaxed (5.3) to (5.4), we may relax the minimization problem (5.6) to

$$\min_{H \in \mathbb{R}^{n \times k}} \operatorname{Tr}(H^T L H) \text{ subject to } H^T H = I_k \text{ and } F^T H = 0_{(h-1)\times k}. \tag{5.7}$$

Our proposed approach to incorporate the fairness notion by [28] into the SC framework consists of solving (5.7) instead of (5.4) (and, as before, applying $k$-means clustering to the rows of an optimal $H$ in order to infer a clustering). Our approach is analogous to the numerous versions of constrained SC that try to incorporate must-link constraints ("vertices A and B should end up in the same cluster") by putting a linear constraint on $H$ (e.g., [101, 102]; see Section 5.5).

Next, we describe a straightforward way to solve (5.7), which is also discussed by [101].

It is easy to see that $\text{rank}(F) = \text{rank}(F^T) = h - 1$. We need to assume that $k \leq n - h + 1$ since otherwise (5.7) does not have any solution. Let $Z \in \mathbb{R}^{n \times (n-h+1)}$ be a matrix whose columns form an orthonormal basis of the nullspace of $F^T$. We can substitute $H = ZY$ for $Y \in \mathbb{R}^{(n-h+1) \times k}$ and then, using that $Z^T Z = I_{(n-h+1)}$, problem (5.7) becomes

$$\min_{Y \in \mathbb{R}^{(n-h+1) \times k}} \text{Tr}(Y^T Z^T L Z Y) \text{ subj. to } Y^T Y = I_k. \tag{5.8}$$

Similarly to problem (5.4), a solution to (5.8) is given by the matrix $Y$ that contains some orthonormal eigenvectors corresponding to the $k$ smallest eigenvalues (respecting multiplicities) of $Z^T L Z$ as columns. We then set $H = ZY$.

This way of solving (5.7) gives rise to our "fair" version of unnormalized SC as stated in Algorithm 12. Note that just as there is no guarantee on the RatioCut value of the output of Algorithm 11 or Algorithm 12 compared to an optimal clustering, in general, there is also no guarantee on how fair the output of Algorithm 12 is. We will still refer to Algorithm 12 as our *fair* version of SC. Similarly to how we proceeded here, in Section 5.7, we incorporate the fairness constraints into normalized SC and state our fair version of normalized SC as Algorithm 13.

**Computational complexity** We will provide a complete discussion of the complexity of our algorithms in Section 5.8. With the implementation as stated, the complexity of both Algorithm 12 and Algorithm 13 is $\mathcal{O}(n^3)$ regarding time and $\mathcal{O}(n^2)$ regarding space, which is the same as the worst-case complexity of standard SC when the number of clusters can be arbitrary. One could apply one of the techniques suggested in the existing literature on constrained spectral clustering to speed up computation (e.g., [101], or [103]; see Section 5.5), but most of these techniques only work for $k = 2$ clusters.

## 5.4 A variant of the stochastic block model

In this section, our goal is to model data sets that have two or more meaningful ground-truth clusterings, of which only one is fair, and show that our algorithms recover the fair ground-truth clustering. If there was only one meaningful ground-truth clustering and this clustering was fair, then any clustering algorithm that is able to recover the ground-truth clustering (e.g., standard SC) would be a fair algorithm. To this end, we define a variant of the famous stochastic block model [SBM; 104]. The SBM is a random graph model that has been widely used to study the performance of clustering algorithms, including standard SC (see Section 5.5 for related work). In the traditional SBM there is a ground-truth clustering of the vertex set $V = [n]$ into $k$ clusters, and in a random graph generated from the model, two vertices $i$ and $j$ are connected with a probability that only depends on which clusters $i$ and $j$ belong to.

In our variant of the SBM we assume that $V = [n]$ comprises $h$ groups $V = V_1 \dot\cup \ldots \dot\cup V_h$ and is partitioned into $k$ ground-truth clusters $V = C_1 \dot\cup \ldots \dot\cup C_k$ such that $|V_s \cap C_l|/|C_l| = \eta_s$, $s \in [h], l \in [k]$, for some $\eta_1, \ldots, \eta_h \in (0,1)$ with $\sum_{s=1}^{h} \eta_s = 1$. Hence, in every cluster each group is represented with the same fraction as in the whole data set $V$ and this ground-truth clustering is fair. Now we define a random graph on $V$ by connecting two vertices $i$ and $j$ with a certain probability $\Pr(i,j)$ that only depends on whether $i$ and $j$ are in the same cluster (or not) and on whether $i$ and $j$ are in the same group (or not). More

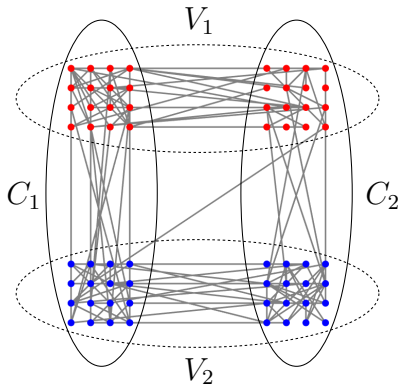



Figure 5.1: Example of a graph generated from our variant of the SBM. There are two meaningful ground-truth clusterings into two clusters: $V = C_1 \dot{\cup} C_2$ and $V = V_1 \dot{\cup} V_2$. Only the first one is fair.

specifically, we have

$$\Pr(i,j) = \begin{cases} a, & i \text{ and } j \text{ in same cluster and in same group,} \\ b, & i \text{ and } j \text{ not in same cluster, but in same group,} \\ c, & i \text{ and } j \text{ in same cluster, but not in same group,} \\ d, & i \text{ and } j \text{ not in same cluster and not in same group,} \end{cases} \tag{5.9}$$

and assume that $a > b > c > d$. As in the ordinary SBM, connecting $i$ and $j$ is independent of connecting $i'$ and $j'$ for $\{i,j\} \neq \{i',j'\}$. Every edge is assigned a weight of $+1$, that is no two connected vertices are considered more similar to each other than any two other connected vertices.

An example of a graph generated from our model (with $h = k = 2$ and $\eta_1 = \eta_2 = 1/2$) can be seen in Figure 5.1. We can see that there are two meaningful ground-truth clusterings into two clusters: $V = C_1 \dot{\cup} C_2$ and $V = V_1 \dot{\cup} V_2$. Among these two clusterings, only $V = C_1 \dot{\cup} C_2$ is fair since $\mathrm{balance}(C_i) = 1$ while $\mathrm{balance}(V_i) = 0$ for $i \in \{1,2\}$. Note that the clustering $V = V_1 \dot{\cup} V_2$ has a smaller RatioCut value than $V = C_1 \dot{\cup} C_2$ because there are more edges between $V_s \cap C_1$ and $V_s \cap C_2$ ($s = 1$ or $s = 2$) than between $V_1 \cap C_l$ and

$V_2 \cap C_l$ ($l = 1$ or $l = 2$). As we will see in the experiments in Section 5.6 (and can also be seen from the proof of the following Theorem 5.4.1), for such a graph, standard SC is very likely to return the unfair clustering $V = V_1 \dot{\cup} V_2$ as output. In contrast, our fair versions of SC return the fair clustering $V = C_1 \dot{\cup} C_2$ with high probability (proof is in section 5.9).

**Theorem 5.4.1** (fair SC succeeds on variant of stochastic block model)**.** *Let $V = [n]$ comprise $h = h(n)$ groups $V = V_1 \dot{\cup} \ldots \dot{\cup} V_h$ and be partitioned into $k = k(n)$ ground-truth clusters $V = C_1 \dot{\cup} \ldots \dot{\cup} C_k$ such that for all $s \in [h]$ and $l \in [k]$*

$$|V_s| = \frac{n}{h}, \quad |C_l| = \frac{n}{k}, \quad \frac{|V_s \cap C_l|}{|C_l|} = \frac{1}{h}. \tag{5.10}$$

*Let $G$ be a random graph constructed according to our variant of the stochastic block model* (5.9) *with probabilities $a = a(n)$, $b = b(n)$, $c = c(n)$, $d = d(n)$ satisfying $a > b > c > d$ and $a \geq C \ln n/n$ for some $C > 0$.*

*Assume that we run Algorithm 12 or Algorithm 13 (stated in Section 5.7) on $G$, where we apply a $(1 + M)$-approximation algorithm to the $k$-means problem encountered in the last step of Algorithm 12 or Algorithm 13, for some $M > 0$. Then, for every $r > 0$, there exist constants $\widehat{C}_i = \widehat{C}_i(C, r)$ and $\widetilde{C}_i = \widetilde{C}_i(C, r)$, $i \in \{1, 2\}$, such that the following is true:*

- ***Unnormalized SC with fairness constraints***

  *If*

$$\frac{a \cdot k^3 \cdot \ln n}{(c - d)^2 \cdot n} < \frac{\widehat{C}_1}{1 + M}, \tag{5.11}$$

  *then with probability at least $1 - n^{-r}$, the clustering returned by Algorithm 12 misclassifies at most*

$$\widetilde{C}_1 \cdot (1 + M) \cdot \frac{a \cdot k^2 \cdot \ln n}{(c - d)^2} \tag{5.12}$$

*many vertices.*

- ***Normalized SC with fairness constraints***

  *Let* $\lambda_1 = \frac{n}{kh}(a + (h-1)c) + \frac{n(k-1)}{kh}(b + (h-1)d)$.

  *If*

$$\frac{\sqrt{k \cdot a \cdot n \ln n}}{\lambda_1 - a} < \frac{\widehat{C}_2}{1+M} \text{ and } \frac{a \cdot k^4 \cdot \ln n}{(c-d)^2 \cdot n} < \frac{\widehat{C}_2}{1+M}, \tag{5.13}$$

  *then with probability at least* $1 - n^{-r}$*, the clustering returned by Algorithm 13 misclassifies at most*

$$\widetilde{C}_2 \cdot (1+M) \cdot \left[ \frac{a \cdot k^3 \cdot \ln n}{(c-d)^2} + \frac{a \cdot n^2 \cdot \ln n}{(\lambda_1 - a)^2} \right] \tag{5.14}$$

  *many vertices.*

We make several remarks on Theorem 5.4.1:

1. By "misclassifies at most $x$ many vertices" we mean that, considering the index $l$ of the cluster $C_l$ that a vertex belongs to as the vertex's class label, there exists a permutation of cluster indices $1, \ldots, k$ such that up to this permutation the clustering returned by our algorithm predicts the correct class label for all but $x$ many vertices.

2. The condition (5.11) is satisfied, for $n$ sufficiently large and assuming that $M \in \mathcal{O}(\ln k)$ (see the next remark), in various regimes: assuming that $k \in \mathcal{O}(n^s)$ for some $s \in [0, 1/3)$, it is satisfied in the dense regime $a, b, c, d \sim \text{const}$, but also in the sparse regime $a, b, c, d \sim \text{const} \cdot (\ln n / n)^q$ for some $q \in [0, 1-3s)$.

   The same is true for condition (5.13), but here we require $s \in [0, 1/4)$ and $q \in [0, 1-4s)$. We suspect that condition (5.13), with respect to $k$, is stronger than necessary. We also suspect that the error bound in (5.14) is not tight with respect to $k$. Note that

in (5.14), both in the dense and in the sparse regime, the term $a \cdot k^3 \cdot \ln n/(c-d)^2$ is dominating over the term $a \cdot n^2 \cdot \ln n/(\lambda_1 - a)^2$ by the factor $k^3$.

Both in the dense and in the sparse regime, under these assumptions on $s$, $q$ and $M$, the error bounds (5.12) and (5.14) divided by $n$, that is the fraction of misclassified vertices, tends to zero as $n$ goes to infinity. Using the terminology prevalent in the literature on community detection in SBMs (see Section 5.5), we may say that our algorithms are *weakly consistent* or solve the *almost exact recovery* problem.

3. There are efficient approximation algorithms for the $k$-means problem in $\mathbb{R}^l$. An algorithm by [105] achieves a constant approximation factor and has running time polynomial in $n$, $k$ and $l$, where $n$ is the number of data points. There is also the famous $(1+\varepsilon)$-approximation algorithm by [106] with running time linear in $n$ and $l$, but exponential in $k$ and $1/\varepsilon$. The algorithm most widely used in practice (e.g., as default method in MATLAB) is $k$-means++, which is a randomized $\mathcal{O}(\ln k)$-approximation algorithm [107].

4. We show empirically in Section 5.6 that our algorithms are also able to find the fair ground-truth clustering in a graph constructed according to our variant of the SBM when (5.10) is not satisfied, that is when the clusters are of different size or the balance of the fair ground-truth clustering is smaller than 1 (i.e. $\eta_s \neq 1/h$ for some $s \in [h]$). For Algorithm 13 the violation of (5.10) can be more severe than for Algorithm 12. In general, we observe Algorithm 13 to outperform Algorithm 12. This is in accordance with standard SC, for which normalized SC has been observed to outperform unnormalized SC [97, 108].

The proof of Theorem 5.4.1 can be found in Section 5.9. It consists of two technical challenges (described here only for the unnormalized case). The first one is to compute the eigenvalues and eigenvectors of the matrix $Z^T \mathcal{L} Z$, where $\mathcal{L}$ is the expected Laplacian matrix of the random graph $G$ and $Z$ is the matrix computed in Algorithm 12. Let $\mathcal{Y}$

be a matrix containing some orthonormal eigenvectors corresponding to the $k$ smallest eigenvalues of $Z^T \mathcal{L} Z$ as columns and $Y$ be a matrix containing orthonormal eigenvectors corresponding to the $k$ smallest eigenvalues of $Z^T L Z$, where $L$ is the observed Laplacian matrix of $G$. The second challenge is to prove that with high probability, $ZY$ is close to $Z\mathcal{Y}$. For doing so we make use of the famous Davis-Kahan sin$\Theta$ Theorem [109]. After that, we can use existing results about $k$-means clustering of perturbed eigenvectors [110] to derive the theorem.

## 5.5 Related work

**Spectral clustering and stochastic block model** SC is one of the most prominent clustering techniques, with a long history and an abundance of related papers. See [97] or [111] for general introductions and an overview of the literature. There are numerous papers on constrained SC, where the goal is to incorporate prior knowledge about the target clustering (usually in the form of must-link and/or cannot-link constraints) into the SC framework [e.g., 112, 101, 113, 114, 103, 115, 116, 117, 102, 118, 119, 120]. Most of these papers are motivated by the use of SC in image or video segmentation.

Closely related to our work are the papers by [101, 103, 116, 102], which incorporate the prior knowledge by imposing a linear constraint in the RatioCut or NCut optimization problem analogous to how we derived our fair versions of SC. These papers provide efficient algorithms to solve the resulting optimization problems. However, the iterative algorithms by [103, 116, 102] only work for $k = 2$ clusters. The method by [101] works for arbitrary $k$ and could be used to speed up the computation of a solution of (5.7) or (5.18) compared to our straightforward way as implemented by Algorithm 12 and Algorithm 13, respectively, but requires to modify the eigensolver in use.

The stochastic block model [SBM; 104] is the canonical model to study the performance of clustering algorithms. There exist several variants of the original model such as the degree-corrected SBM or the labeled SBM. For a recent survey see [121]. In the

labeled SBM, vertices can carry a label that is correlated with the ground-truth clustering. This is quite the opposite of our model, in which the group-membership information is "orthogonal" to the ground-truth clustering. Several papers show the consistency (i.e., the capability to recover the ground-truth clustering) of different versions of SC on the SBM or the degree-corrected SBM under different assumptions [122, 123, 124, 110, 125, 126].

For example, [122] show consistency of normalized SC assuming that the minimum expected vertex degree is in $\Omega(n/\sqrt{\log n})$, while [110] show that SC based on the adjacency matrix is consistent requiring only that the maximum expected degree is in $\Omega(\sqrt{\log n})$. Note that these papers also make assumptions on the eigenvalues of the expected Laplacian or adjacency matrix while all assumptions and guarantees stated in our Theorem 5.4.1 directly depend on the connection probabilities $a, b, c, d$ of our model. We are not aware of any work providing consistency results for constrained SC methods as we do in this chapter.

**Fairness** By now, there is a huge body of work on fairness in machine learning. For a recent paper providing an overview of the literature on fair classification see [127]. Our work adds to the literature on fair methods for unsupervised learning tasks [28, 128, 95, 129, 96, 30]. Note that all these papers assume to know which demographic group a data point belongs to just as we do. We discuss the pieces of work most closely related to our work.

[28] proposed the notion of fairness for clustering underlying our work. It is based on the fairness notion of disparate impact [130] and the $p\%$-rule [131], respectively, which essentially say that the output of a machine learning algorithm should be independent of a sensitive attribute. In their paper, [28] focus on $k$-median and $k$-center clustering. For the case of a binary sensitive attribute, that is there are only two demographic groups, they provide approximation algorithms for the problems of finding a clustering with minimum $k$-median / $k$-center cost under the constraint that all clusters have some prespecified level of balance. Subsequently, [29] provide an approximation algorithm for such a fair $k$-center

problem with multiple groups. [30] build upon the fairness notion and techniques of [28] and devise an approximation algorithm for the fair $k$-means problem, assuming that there are only two groups of the same size.

## 5.6 Experiments

In this section, we present a number of experiments. We first study our fair versions of spectral clustering, Algorithm 12 and Algorithm 13, on synthetic data generated according to our variant of the SBM and compare our algorithms to standard SC. We also study how robust our algorithms are with respect to a certain perturbation of our model. We then compare our algorithms to standard SC on real network data. We implemented all algorithms in MATLAB. We used the built-in function for $k$-means clustering with all parameters set to their default values except for the number of replicates, which we set to 10. In Figures 5.2 and 5.3, all plots show average results obtained from running an experiment for 100 times for different parameter settings.

### 5.6.1 Synthetic data

We run experiments on our variant of the SBM introduced in Section 5.4. To asses the quality of a clustering we measure the fraction of misclassified vertices w.r.t. the fair ground-truth clustering (cf. Section 5.4), which we refer to as error.

In the experiments of Figure 5.2 and 5.3, we study the performance of standard unnormalized and normalized SC and of our fair versions, Algorithm 12 and Algorithm 13, as a function of $n$. Due to the high running time of Algorithm 13 (cf. Section 5.3), we only run it up to $n = 4000$. All plots show the error of the methods, except for the fourth plot in Figure 5.2, which shows their run-time. We study several parameter settings. For the plots in the Figure 5.2, Assumption (5.10) in Theorem 5.4.1 is satisfied, that is $|V_s \cap C_l| = \frac{n}{kh}$ for all $s \in [h]$ and $l \in [k]$. In this case, in accordance with Theorem 5.4.1, both Algorithm 12 and Algorithm 13 are able to recover the fair ground-truth clustering if $n$ is just

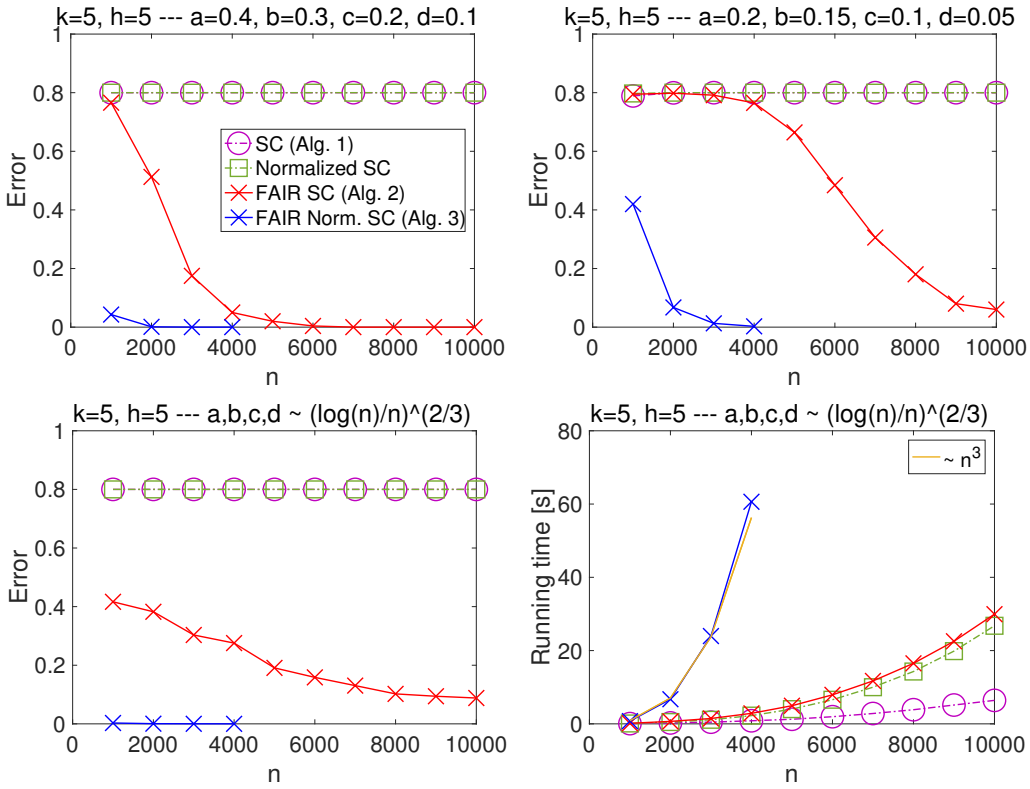



Figure 5.2: Performance of standard spectral clustering and our fair versions on our variant of the stochastic block model as a function of $n$ for various parameter settings. Error is the fraction of misclassified vertices w.r.t. the fair ground-truth clustering (cf. Section 5.4). Assumption (5.10) in Theorem 5.4.1 is satisfied, that is $|V_s \cap C_l| = \frac{n}{kh}$, $s \in [h]$, $l \in [k]$.

large enough while standard SC always fails to do so. Algorithm 13 yields significantly better results than Algorithm 12 and requires much smaller values of $n$ for achieving zero error. This comes at the cost of a higher running time of Algorithm 13 (still it is in $\mathcal{O}(n^3)$ as claimed in Section 5.3). The run-time of Algorithm 12 is the same as the run-time of standard normalized SC. For the plots in the Figure 5.3, Assumption (5.10) in Theorem 5.4.1 is not satisfied. We consider various scenarios of cluster sizes $|C_l|$ and group sizes $|V_s|$ (however, we always have $|V_s \cap C_l|/|C_l| = |V_s|/n$, $s \in [h]$, $l \in [k]$, so that $C_1 \dot{\cup} \ldots \dot{\cup} C_k$ is as fair as possible). When the cluster sizes are different, but the group sizes are all equal to each other (1st plot in the 2nd row) or Assumption (5.10) is only slightly violated (2nd plot), both Algorithm 12 and Algorithm 13 are still able to recover the fair ground-truth clustering. Compared to the plots in the Figure 5.2, Algorithm 12 requires a larger value

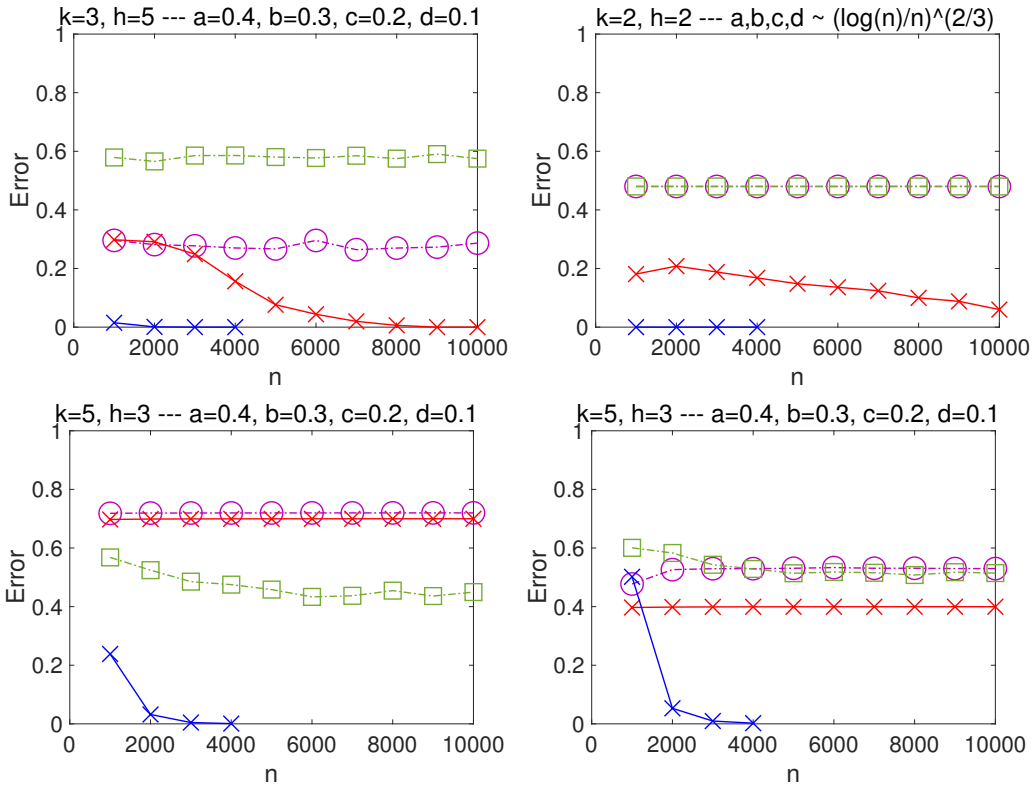



Figure 5.3: Performance of standard spectral clustering and our fair versions on our variant of the stochastic block model as a function of $n$ for various parameter settings. Error is the fraction of misclassified vertices w.r.t. the fair ground-truth clustering (cf. Section 5.4).Assumption (5.10) is not satisfied. **2nd row, 1st plot:** $\frac{|C_1|}{n} = \frac{7}{10}$, $\frac{|C_2|}{n} = \frac{|C_3|}{n} = \frac{15}{100}$ and $\frac{|V_s \cap C_l|}{|C_l|} = \frac{1}{5}$, $s \in [5]$, $l \in [5]$; **2nd plot:** $\frac{|C_1|}{n} = \frac{6}{10}$, $\frac{|C_2|}{n} = \frac{4}{10}$ and $\frac{|V_1 \cap C_l|}{|C_l|} = \frac{6}{10}$, $\frac{|V_2 \cap C_l|}{|C_l|} = \frac{4}{10}$, $l \in [2]$; **3rd plot:** $\frac{|C_1|}{n} = \frac{|C_2|}{n} = \frac{3}{10}$, $\frac{|C_3|}{n} = \frac{2}{10}$, $\frac{|C_4|}{n} = \frac{|C_5|}{n} = \frac{1}{10}$ and $\frac{|V_1 \cap C_l|}{|C_l|} = \frac{5}{10}$, $\frac{|V_2 \cap C_l|}{|C_l|} = \frac{3}{10}$, $\frac{|V_3 \cap C_l|}{|C_l|} = \frac{2}{10}$, $l \in [5]$; **4th plot:** $\frac{|C_1|}{n} = \frac{6}{10}$, $\frac{|C_2|}{n} = \ldots = \frac{|C_5|}{n} = \frac{1}{10}$ and $\frac{|V_1 \cap C_l|}{|C_l|} = \frac{5}{10}$, $\frac{|V_2 \cap C_l|}{|C_l|} = \frac{3}{10}$, $\frac{|V_3 \cap C_l|}{|C_l|} = \frac{2}{10}$, $l \in [5]$.

of $n$ though, even though $k$ is smaller. Algorithm 13 achieves (almost) zero error already for $n = 1000$ in these scenarios. When Assumption (5.10) is strongly violated (3rd and 4th plot), Algorithm 12 fails to recover the fair ground-truth clustering, but Algorithm 13 still succeeds.

In the experiments shown in Figure 5.4, we study the error of Algorithm 12 (left plot) and Algorithm 13 (right plot) as a function of $k$ when $n$ is roughly fixed. More precisely, for $k \in \{2, \ldots, 8\}$ and $h = 4$, we have $n = kh\lceil \frac{5000}{kh} \rceil$ (Alg. 12; left) or $n = kh\lceil \frac{2000}{kh} \rceil$ (Alg. 13; right), which allows for fair ground-truth clusterings satisfying (5.10). We con-

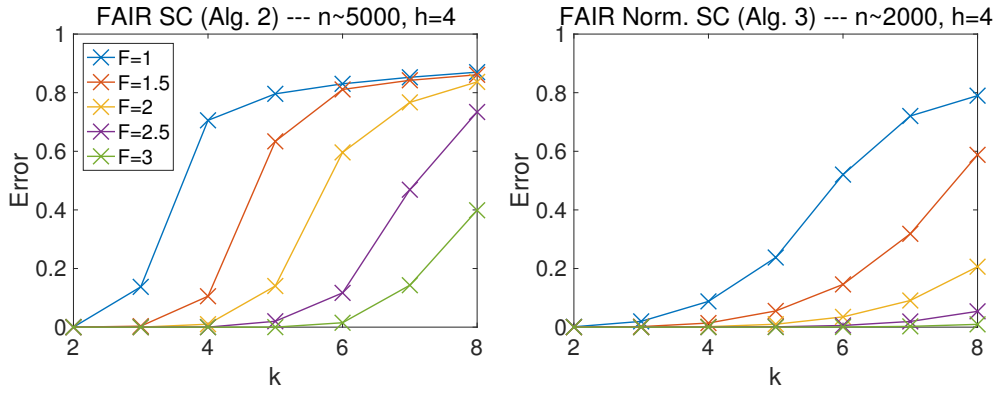



Figure 5.4: Error of our algorithms as a function of $k$. We consider $a = F \cdot \frac{25}{100}$, $b = F \cdot \frac{2}{10}$, $c = F \cdot \frac{15}{100}$, $d = F \cdot \frac{1}{10}$ for various values of $F$. **Left:** Alg. 12, $n \approx 5000$. **Right:** Alg. 13, $n \approx 2000$.

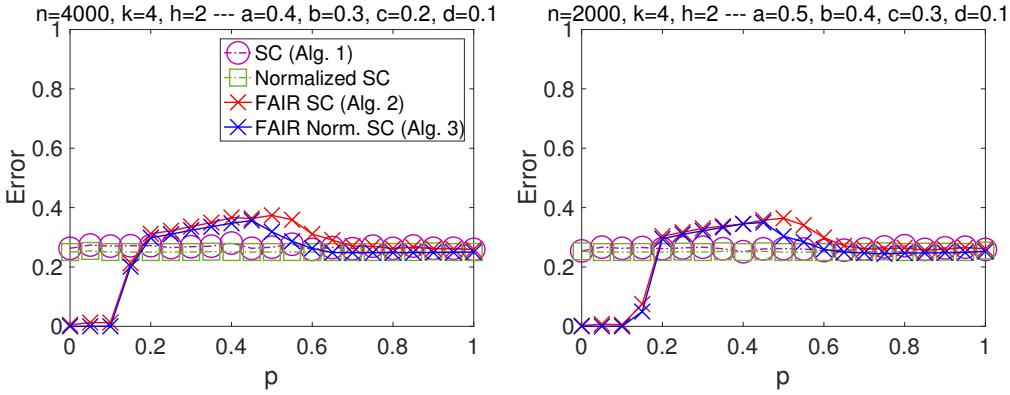



Figure 5.5: Error of standard spectral clustering and our fair versions as a function of the perturbation parameter $p$.

sider connection probabilities $a = F \cdot \frac{25}{100}$, $b = F \cdot \frac{2}{10}$, $c = F \cdot \frac{15}{100}$, $d = F \cdot \frac{1}{10}$ for $F \in \{1, 1.5, 2, 2.5, 3\}$. Unsurprisingly, for both Algorithm 12 and Algorithm 13 the error is monotonically increasing with $k$. The rate of increase critically depends on $F$ (or the probabilities $a, b, c, d$). For Algorithm 12, this is even more severe. There is only a small range in which the various curves exhibit polynomial growth, which makes it impossible to empirically evaluate whether our error guarantees (5.12) and (5.14) are tight with respect to $k$.

In the experiments of Figure 5.5, we consider a perturbation of our model as follows: first, for $n = 4000$ (left plot) or $n = 2000$ (right plot), $k = 4$ and $h = 2$ we generate a graph from our model just as before (Assumption (5.10) is satisfied; in particular, the two groups

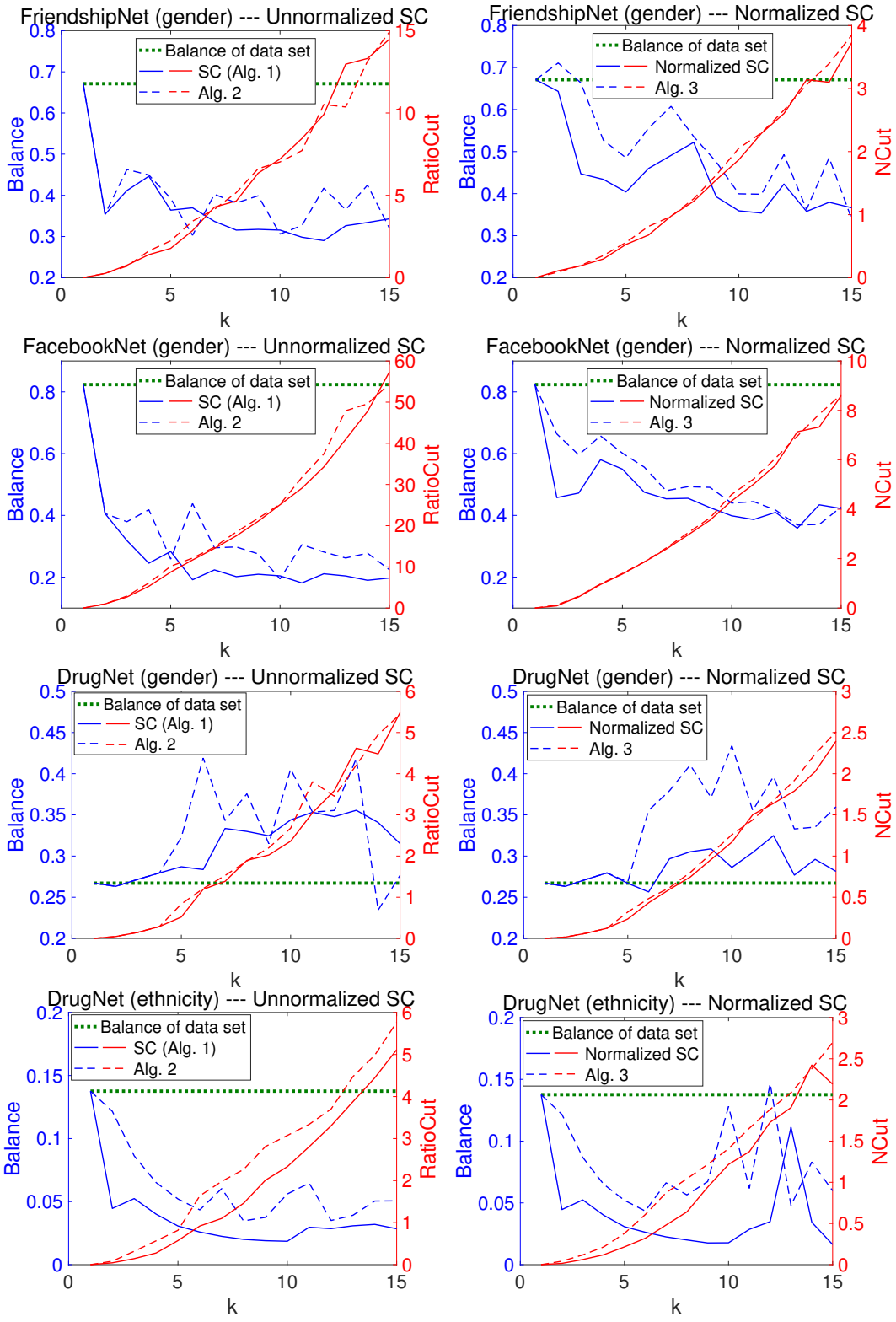



Figure 5.6: Balance (left axis) and RatioCut/NCut value (right axis) of standard SC and our fair versions as a function of $k$ on real networks.

have the same size), but then we assign some of the vertices in the first group to the other group. Concretely, for a perturbation parameter $p \in [0, 1]$, each vertex in the first group is assigned to the second one with probability $p$ independently of each other. The case $p = 0$ is our model without any perturbation. If $p = 1$, there is only one group and our algorithms technically coincide with standard unnormalized or normalized SC. The two plots show the error of our algorithms and standard SC as a function of $p$. Both our algorithms show the same behavior. They are robust against the perturbation up to $p = 0.15$. They yield the same error as standard SC for $p \geq 0.7$.

### 5.6.2 Real data

In the experiments of Figure 5.6, we evaluate the performance of standard unnormalized and normalized SC versus our fair versions on real network data. The quality of a clustering is measured through its "Balance" (defined as the average of the balance (5.5) over all clusters; shown on left axis of the plots) and its RatioCut (5.1) or NCut (5.15) value (right axis). All networks that we are working with are the largest connected component of an originally unconnected network.

The first two rows of Figure 5.6 shows the results as a function of the number of clusters $k$ for two high school friendship networks [132]. Vertices correspond to students and are split into two groups of males and females. FRIENDSHIPNET has 127 vertices and an edge between two students indicates that one of them reported friendship with the other one. FACEBOOKNET consists of 155 vertices and an edge between two students indicates friendship on Facebook. As we can see from the plots, compared to standard SC, our fair versions improve the output clustering's balance (by $10\%$ / $15\%$ / $34\%$ / $10\%$ on average over $k$) while almost not changing its RatioCut or NCut value.

The third and fourth rows shows the results for DRUGNET, a network encoding acquaintanceship between drug users in Hartford, CT [133]. In the third row, the network consists of 185 vertices split into two groups of males and females (we had to remove some

vertices for which the gender was not known). In the fourth row, the network has 193 vertices split into three ethnic groups of African Americans, Latinos and others. Again, our fair versions of SC quite significantly improve the balance of the output clustering over standard SC (by $5\%$ / $18\%$ / $86\%$ / $167\%$ on average over $k$). However, in the fourth row we also observe a moderate increase of the RatioCut or NCut value.

### 5.7 Adaptation of normalized spectral clustering to fairness constraint

In this section we derive a fair version of normalized spectral clustering (similarly to how we proceeded for unnormalized spectral clustering in Sections 5.2 and 5.3).

Normalized spectral clustering aims at partitioning $V$ into $k$ clusters with minimum value of the NCut objective function as follows [see 97, for details]: for a clustering $V = C_1 \dot\cup \ldots \dot\cup C_k$ we have

$$\mathrm{NCut}(C_1, \ldots, C_k) = \sum_{l=1}^{k} \frac{\mathrm{Cut}(C_l, V \setminus C_l)}{\mathrm{vol}(C_l)}, \tag{5.15}$$

where $\mathrm{vol}(C_l) = \sum_{i \in C_l} d_i = \sum_{i \in C_l, j \in [n]} W_{ij}$. Encoding a clustering $V = C_1 \dot\cup \ldots \dot\cup C_k$ by a matrix $H \in \mathbb{R}^{n \times k}$ with

$$H_{il} = \begin{cases} 1/\sqrt{\mathrm{vol}(C_l)}, & i \in C_l, \\ 0, & i \notin C_l \end{cases}, \tag{5.16}$$

we have $\mathrm{NCut}(C_1, \ldots, C_k) = \mathrm{Tr}(H^T L H)$. Note that any $H$ of the form (5.16) satisfies $H^T D H = I_k$. Normalized spectral clustering relaxes the problem of minimizing $\mathrm{Tr}(H^T L H)$ over all $H$ of the form (5.16) to

$$\min_{H \in \mathbb{R}^{n \times k}} \mathrm{Tr}(H^T L H) \text{ subject to } H^T D H = I_k. \tag{5.17}$$

Substituting $H = D^{-1/2} T$ for $T \in \mathbb{R}^{n \times k}$ (we need to assume that $G$ does not contain any

isolated vertices since otherwise $D^{-1/2}$ does not exist), problem (5.17) becomes

$$\min_{T \in \mathbb{R}^{n \times k}} \operatorname{Tr}(T^T D^{-1/2} L D^{-1/2} T) \text{ subject to } T^T T = I_k.$$

Similarly to unnormalized spectral clustering, normalized spectral clustering computes an optimal $T$ by computing the $k$ smallest eigenvalues and some corresponding eigenvectors of $D^{-1/2}LD^{-1/2}$ and applies $k$-means clustering to the rows of $H = D^{-1/2}T$ (in practice, $H$ can be computed directly by solving the generalized eigenproblem $Lx = \lambda Dx$, $x \in \mathbb{R}^n$, $\lambda \in \mathbb{R}$; see [97]).

Now we want to derive our fair version of normalized spectral clustering. The first step is to show that Lemma 5.3.1 holds true if we encode a clustering as in (5.16):

**Lemma 5.7.1** (Fairness constraint as linear constraint on $H$ for normalized spectral clustering)**.** *For $s \in [h]$, let $f^{(s)} \in \{0,1\}^n$ be the group-membership vector of $V_s$, that is $f_i^{(s)} = 1$ if $i \in V_s$ and $f_i^{(s)} = 0$ otherwise. Let $V = C_1 \dot\cup \ldots \dot\cup C_k$ be a clustering that is encoded as in* (5.16)*. We have, for every $l \in [k]$,*

$$\forall s \in [h-1] : \sum_{i=1}^{n} \left( f_i^{(s)} - \frac{|V_s|}{n} \right) H_{il} = 0 \quad \Leftrightarrow \quad \forall s \in [h] : \frac{|V_s \cap C_l|}{|C_l|} = \frac{|V_s|}{n}.$$

*Proof.* This simply follows from

$$\sum_{i=1}^{n} \left( f_i^{(s)} - \frac{|V_s|}{n} \right) H_{il} = \frac{|V_s \cap C_l|}{\sqrt{\operatorname{vol}(C_l)}} - \frac{|V_s| \cdot |C_l|}{n\sqrt{\operatorname{vol}(C_l)}}$$

and $|C_l| = \sum_{s=1}^{h} |V_s \cap C_l|$. □

Lemma 5.7.1 suggests that in a fair version of normalized spectral clustering, rather than solving (5.17), we should solve

$$\min_{H \in \mathbb{R}^{n \times k}} \operatorname{Tr}(H^T L H) \text{ subject to } H^T D H = I_k \text{ and } F^T H = 0_{(h-1)\times k}, \tag{5.18}$$

**Algorithm 13:** Normalized SC with fairness constraints

**Input:** weighted adjacency matrix $W \in \mathbb{R}^{n \times n}$
**(the underlying graph must not contain any isolated vertices)**; $k \in \mathbb{N}$; group-membership vectors $f^{(s)} \in \{0,1\}^n$, $s \in [h]$

**Output:** a clustering of $[n]$ into $k$ clusters

- compute the Laplacian matrix $L = D - W$ with the degree matrix $D$
- build the matrix $F$ that has the vectors $f^{(s)} - \frac{|V_s|}{n} \cdot \mathbf{1}_n$, $s \in [h-1]$, as columns
- compute a matrix $Z$ whose columns form an orthonormal basis of the nullspace of $F^T$
- compute the square root $Q$ of $Z^T D Z$
- compute some orthonormal eigenvectors corresponding to the $k$ smallest eigenvalues (respecting multiplicities) of $Q^{-1} Z^T L Z Q^{-1}$
- let $X$ be a matrix containing these eigenvectors as columns
- apply $k$-means clustering to the rows of $H = ZQ^{-1}X \in \mathbb{R}^{n \times k}$, which yields a clustering of $[n]$ into $k$ clusters

where $F \in \mathbb{R}^{n \times (h-1)}$ is the matrix that has the vectors $f^{(s)} - (|V_s|/n) \cdot \mathbf{1}_n$, $s \in [h-1]$, as columns (just as in Section 5.3). It is $\text{rank}(F) = \text{rank}(F^T) = h - 1$ and we need to assume that $k \leq n - h + 1$ since otherwise (5.18) does not have any solution. Let $Z \in \mathbb{R}^{n \times (n-h+1)}$ be a matrix whose columns form an orthonormal basis of the nullspace of $F^T$. We substitute $H = ZY$ for $Y \in \mathbb{R}^{(n-h+1) \times k}$ and then problem (5.18) becomes

$$\min_{Y \in \mathbb{R}^{(n-h+1) \times k}} \text{Tr}(Y^T Z^T L Z Y) \text{ subject to } Y^T Z^T D Z Y = I_k. \tag{5.19}$$

Assuming that $G$ does not contain any isolated vertices, $Z^T D Z$ is positive definite and hence has a positive definite square root, that is there exists a positive definite $Q \in \mathbb{R}^{(n-h+1) \times (n-h+1)}$ with $Z^T D Z = Q^2$. We can substitute $Y = Q^{-1}X$ for $X \in \mathbb{R}^{(n-h+1) \times k}$ and then problem (5.19) becomes

$$\min_{X \in \mathbb{R}^{(n-h+1) \times k}} \text{Tr}(X^T Q^{-1} Z^T L Z Q^{-1} X) \text{ subject to } X^T X = I_k. \tag{5.20}$$

A solution to (5.20) is given by the matrix $X$ that contains some orthonormal eigenvectors corresponding to the $k$ smallest eigenvalues (respecting multiplicities) of $Q^{-1}Z^TLZQ^{-1}$ as columns. This gives rise to our fair version of normalized spectral clustering as stated in Algorithm 13.

## 5.8 Computational complexity of our algorithms

The costs of standard spectral clustering (e.g., Algorithm 11) are dominated by the complexity of the eigenvector computations and are commonly stated to be, in general, in $\mathcal{O}(n^3)$ regarding time and $\mathcal{O}(n^2)$ regarding space for an arbitrary number of clusters $k$, unless approximations are applied [134, 135]. In addition to the computations performed in Algorithm 11, in Algorithm 12 and Algorithm 13 we have to compute an orthonormal basis of the nullspace of $F^T$, perform some matrix multiplications, and (only for Algorithm 13) compute the square root of an $(n-h+1)\times(n-h+1)$-matrix and the inverse of this square root. All these computations can be done in $\mathcal{O}(n^3)$ regarding time and $\mathcal{O}(n^2)$ regarding space (an orthonormal basis of the nullspace of $F^T$ can be computed by means of an SVD; see, e.g., [136]), and hence our algorithms have the same worst-case complexity as standard spectral clustering. On the other hand, if the graph $G$, and thus the Laplacian matrix $L$, is sparse or $k$ is small, then the eigenvector computations in Algorithm 11 can be done more efficiently than with cubic running time [137]. This is not the case for our algorithms as stated. However, one could apply one of the techniques suggested in the existing literature on constrained spectral clustering to speed up computation (e.g., [101], or [103]; see Section 5.5). With the implementation as stated, in our experiments in Section 5.6 we observe that Algorithm 12 has a similar running time as standard normalized spectral clustering while the running time of Algorithm 13 is significantly higher.

## 5.9 Proof of Theorem 5.4.1

We split the proof of Theorem 5.4.1 into four parts. In the first part, we analyze the eigenvalues and eigenvectors of the expected adjacency matrix $\mathcal{W}$ and of the matrix $Z^T\mathcal{L}Z$, where $\mathcal{L}$ is the expected Laplacian matrix and $Z$ is the matrix computed in the execution of Algorithm 12 or Algorithm 13. In the second part, we study the deviation of the observed matrix $Z^TLZ$ from the expected matrix $Z^T\mathcal{L}Z$. In the third part, we use the results from the first and the second part to prove Theorem 5.4.1 for Algorithm 12 (unnormalized SC with fairness constraints). In the fourth part, we prove Theorem 5.4.1 for Algorithm 13 (normalized SC with fairness constraints).

**Notation** For $x \in \mathbb{R}^n$, by $\|x\|$ we denote the Euclidean norm of $x$, that is $\|x\| = \sqrt{x_1^2 + \ldots + x_n^2}$. For $A \in \mathbb{R}^{n\times m}$, by $\|A\|$ we denote the operator norm (also known as spectral norm) and by $\|A\|_F$ the Frobenius norm of $A$. It is

$$\|A\| = \max_{x\in\mathbb{R}^m:\|x\|=1} \|Ax\| = \sqrt{\lambda_{\max}(A^TA)}, \tag{5.21}$$

where $\lambda_{\max}(A^TA)$ is the largest eigenvalue of $A^TA$, and

$$\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} A_{ij}^2} = \sqrt{\mathrm{Tr}(A^TA)}. \tag{5.22}$$

Note that for a symmetric matrix $A \in \mathbb{R}^{n\times n}$ with $A = A^T$ we have $\|A\| = \max\{|\lambda_i| : \lambda_i \text{ is an eigenvalue of } A\}$. It follows from (5.21) and (5.22) that for any $A \in \mathbb{R}^{n\times m}$ with rank at most $r$ we have

$$\|A\| \le \|A\|_F \le \sqrt{r}\|A\|. \tag{5.23}$$

We use $\mathrm{const}(X)$ to denote a universal constant that only depends on $X$ and that may change from line to line.

**Part 1: Eigenvalues and eigenvectors of $\mathcal{W}$ and of $Z^T\mathcal{L}Z$**

Assuming the $n$ vertices $1,\dots,n$ are sorted in a way such that

$$\begin{aligned}
1,\dots,\tfrac{n}{kh} \in C_1\cap V_1,\ \tfrac{n}{kh}+1,\dots,\tfrac{2n}{kh}\in C_1\cap V_2,\dots,\tfrac{(h-1)n}{kh}+1,\dots,\tfrac{n}{k}\in C_1\cap V_h,\\
\tfrac{n}{k}+1,\dots,\tfrac{n}{k}+\tfrac{n}{kh}\in C_2\cap V_1,\dots,\tfrac{n}{k}+\tfrac{(h-1)n}{kh}+1,\dots,\tfrac{2n}{k}\in C_2\cap V_h,\\
\vdots\\
\tfrac{(k-1)n}{k}+1,\dots,\tfrac{(k-1)n}{k}+\tfrac{n}{kh}\in C_k\cap V_1,\dots,\tfrac{(k-1)n}{k}+\tfrac{(h-1)n}{kh}+1,\dots,n\in C_k\cap V_h,
\end{aligned} \tag{5.24}$$

the expected adjacency matrix $\mathcal{W}\in\mathbb{R}^{n\times n}$ is given by the block matrix

$$\mathcal{W} = \underbrace{\begin{pmatrix}
[R] & [S] & [S] & [S] & \cdots & [S] & [S]\\
[S] & [R] & [S] & [S] & \cdots & [S] & [S]\\
[S] & [S] & [R] & [S] & \cdots & [S] & [S]\\
\vdots & & \ddots & & & \vdots & \\
[S] & [S] & [S] & [S] & \cdots & [S] & [R]
\end{pmatrix}}_{=:\widetilde{\mathcal{W}}} - aI_n, \tag{5.25}$$

where $[R] \in \{a, c\}^{\frac{n}{k} \times \frac{n}{k}}$ and $[S] \in \{b, d\}^{\frac{n}{k} \times \frac{n}{k}}$ are themselves block matrices

$$[R] = \begin{pmatrix} [a] & [c] & [c] & [c] & \cdots & [c] & [c] \\ [c] & [a] & [c] & [c] & \cdots & [c] & [c] \\ [c] & [c] & [a] & [c] & \cdots & [c] & [c] \\ & \vdots & & \ddots & & \vdots & \\ [c] & [c] & [c] & [c] & \cdots & [c] & [a] \end{pmatrix}, [S] = \begin{pmatrix} [b] & [d] & [d] & [d] & \cdots & [d] & [d] \\ [d] & [b] & [d] & [d] & \cdots & [d] & [d] \\ [d] & [d] & [b] & [d] & \cdots & [d] & [d] \\ & \vdots & & \ddots & & \vdots & \\ [d] & [d] & [d] & [d] & \cdots & [d] & [b] \end{pmatrix}$$

with $[a]$, $[b]$, $[c]$ and $[d]$ being matrices of size $\frac{n}{kh} \times \frac{n}{kh}$ with all entries equaling $a$, $b$, $c$ and $d$, respectively. We denote the matrix $\mathcal{W} + aI_n$ with $\mathcal{W}$ as in (5.25) by $\widetilde{\mathcal{W}}$. If the vertices are not ordered as in (5.24), the expected adjacency matrix $\mathcal{W}$ is rather given by $\mathcal{W} = P^T \widetilde{\mathcal{W}} P - aI_n$ for some permutation matrix $P \in \{0, 1\}^{n \times n}$ with $PP^T = P^T P = I_n$. Note that $v \in \mathbb{R}^n$ is an eigenvector of $\widetilde{\mathcal{W}}$ with eigenvalue $\lambda$ if and only if $P^T v$ is an eigenvector of $P^T \widetilde{\mathcal{W}} P$ with eigenvalue $\lambda$. Keeping track of the permutation matrices $P$ and $P^T$ throughout the proof of Theorem 5.4.1 does not impose any technical challenges, but makes the writing more complicated. Hence, for simplicity and without loss of generality, we assume in the following that the vertices are ordered as in (5.24).

The following lemma characterizes the eigenvalues and eigenvectors of $\widetilde{\mathcal{W}}$. Clearly, this also characterizes the eigenvalues and eigenvectors of $\mathcal{W}$: $v \in \mathbb{R}^n$ is an eigenvector of $\widetilde{\mathcal{W}}$ with eigenvalue $\lambda$ if and only if $v$ is an eigenvector of $\mathcal{W}$ with eigenvalue $\lambda - a$.

**Lemma 5.9.1.** *Assuming that $a > b > c > d \geq 0$, the matrix $\widetilde{\mathcal{W}}$ has rank $kh$ or rank $k + h - 1$ (the latter is true if and only if $a - c = b - d$). It has the following eigenvalues*

$\lambda_1, \ldots, \lambda_n$:

$$\begin{aligned}
\lambda_1 &= \frac{n}{kh}\left[\big(a+(h-1)c\big)+(k-1)\big(b+(h-1)d\big)\right], \\
\lambda_2 = \lambda_3 = \ldots = \lambda_h &= \frac{n}{kh}\left[\big(a-c\big)+(k-1)\big(b-d\big)\right], \\
\lambda_{h+1} = \lambda_{h+2} = \ldots = \lambda_{h+k-1} &= \frac{n}{kh}\left[\big(a+(h-1)c\big)-\big(b+(h-1)d\big)\right], \\
\lambda_{h+k}, \lambda_{h+k+1} = \ldots = \lambda_{hk} &= \frac{n}{kh}\left[\big(a-c\big)-\big(b-d\big)\right], \\
\lambda_{hk+1} = \lambda_{hk+2} = \ldots = \lambda_n &= 0.
\end{aligned} \tag{5.26}$$

*It is* $\lambda_1 > \lambda_2 = \ldots = \lambda_h > 0$ *as well as* $\lambda_1 > \lambda_{h+1} = \ldots = \lambda_{h+k-1} > 0$ *and* $\lambda_2 = \ldots = \lambda_h > |\lambda_{h+k}| = \ldots = |\lambda_{hk}|$ *as well as* $\lambda_{h+1} = \ldots = \lambda_{h+k-1} > |\lambda_{h+k}| = \ldots = |\lambda_{hk}|$.

*An eigenvector corresponding to* $\lambda_1$ *is given by* $v_1 = \mathbf{1}_n$. *The eigenspace corresponding*

*to the eigenvalue $\lambda_2 = \ldots = \lambda_h$ is spanned by the vectors $v_2, \ldots, v_h$ with*

$$v_2 = \begin{pmatrix} [1] \\ [-\frac{1}{h-1}] \\ [-\frac{1}{h-1}] \\ [-\frac{1}{h-1}] \\ \vdots \\ [-\frac{1}{h-1}] \\ \\ [1] \\ [-\frac{1}{h-1}] \\ [-\frac{1}{h-1}] \\ [-\frac{1}{h-1}] \\ \vdots \\ [-\frac{1}{h-1}] \\ \vdots \\ [1] \\ [-\frac{1}{h-1}] \\ [-\frac{1}{h-1}] \\ [-\frac{1}{h-1}] \\ \vdots \\ [-\frac{1}{h-1}] \end{pmatrix}, \quad v_3 = \begin{pmatrix} [-\frac{1}{h-1}] \\ [1] \\ [-\frac{1}{h-1}] \\ [-\frac{1}{h-1}] \\ \vdots \\ [-\frac{1}{h-1}] \\ \\ [-\frac{1}{h-1}] \\ [1] \\ [-\frac{1}{h-1}] \\ [-\frac{1}{h-1}] \\ \vdots \\ [-\frac{1}{h-1}] \\ \vdots \\ [-\frac{1}{h-1}] \\ [1] \\ [-\frac{1}{h-1}] \\ [-\frac{1}{h-1}] \\ [-\frac{1}{h-1}] \\ \vdots \\ [-\frac{1}{h-1}] \end{pmatrix}, \quad \cdots, \quad v_h = \begin{pmatrix} [-\frac{1}{h-1}] \\ [-\frac{1}{h-1}] \\ \vdots \\ [-\frac{1}{h-1}] \\ [1] \\ [-\frac{1}{h-1}] \\ \\ [-\frac{1}{h-1}] \\ [-\frac{1}{h-1}] \\ \vdots \\ [-\frac{1}{h-1}] \\ [1] \\ [-\frac{1}{h-1}] \\ \vdots \\ [-\frac{1}{h-1}] \\ [-\frac{1}{h-1}] \\ \vdots \\ [-\frac{1}{h-1}] \\ [1] \\ [-\frac{1}{h-1}] \end{pmatrix},$$

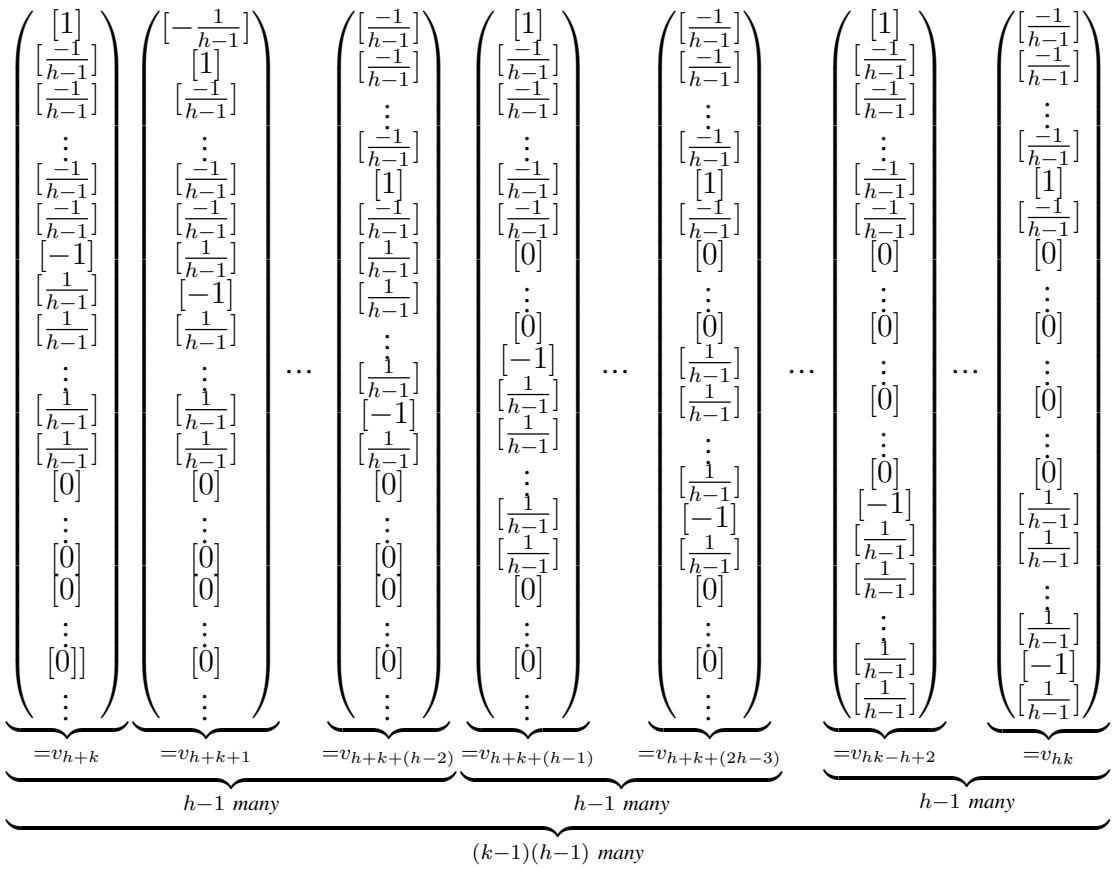
*where for $z \in \mathbb{R}$, by $[z]$ we denote a block of size $\frac{n}{kh}$ with all entries equaling $z$. The eigenspace corresponding to the eigenvalue $\lambda_{h+1} = \ldots = \lambda_{h+k-1}$ is spanned by the vectors*

$v_{h+1}, \ldots, v_{h+k-1}$ *with*

$$v_{h+1} = \begin{pmatrix} [1] \\ [-\frac{1}{k-1}] \\ [-\frac{1}{k-1}] \\ \vdots \\ [-\frac{1}{k-1}] \\ [-\frac{1}{k-1}] \\ [-\frac{1}{k-1}] \end{pmatrix}, \quad v_{h+2} \begin{pmatrix} [-\frac{1}{k-1}] \\ [1] \\ [-\frac{1}{k-1}] \\ \vdots \\ [-\frac{1}{k-1}] \\ [-\frac{1}{k-1}] \\ [-\frac{1}{k-1}] \end{pmatrix}, \quad \cdots, \quad v_{h+k-1} = \begin{pmatrix} [-\frac{1}{k-1}] \\ [-\frac{1}{k-1}] \\ [-\frac{1}{k-1}] \\ \vdots \\ [-\frac{1}{k-1}] \\ [1] \\ [-\frac{1}{k-1}] \end{pmatrix},$$

*where for $z \in \mathbb{R}$, by $[z]$ we denote a block of size $\frac{n}{k}$ with all entries equaling $z$. The eigenspace corresponding to the eigenvalue $\lambda_{h+k} = \ldots = \lambda_{hk}$ is spanned by the vectors*

$v_{h+k}, \ldots, v_{hk}$ *with*




*where for $z \in \mathbb{R}$, by $[z]$ we denote a block of size $\frac{n}{kh}$ with all entries equaling $z$.*

*Proof.* The matrix $\widetilde{\mathcal{W}}$ has only $kh$ different columns and hence rank$\tilde{\mathcal{A}} \leq kh$ and there are at most $kh$ many non-zero eigenvalues. The statement about the rank of $\widetilde{\mathcal{W}}$ follows from the statement about the eigenvalues of $\widetilde{\mathcal{W}}$.

It is easy to verify that any of the vectors $v_i$ is actually an eigenvector of $\widetilde{\mathcal{W}}$ corresponding to eigenvalue $\lambda_i$, $i \in [hk]$. We need to show that the vectors $v_2, \ldots, v_h$, the vectors $v_{h+1}, \ldots, v_{h+k-1}$, as well as the vectors $v_{h+k}, \ldots, v_{hk}$ are linearly independent. For example, let us show that $v_2, \ldots, v_h$ are linearly independent: assume that $\sum_{j\in\{2,\ldots,h\}} \alpha_j v_j = 0$ for some $\alpha_j \in \mathbb{R}$. We need to show that $\alpha_j = 0$, $j \in \{2, \ldots, h\}$. Looking at the $n$-th coordinate of $\sum_{j\in\{2,\ldots,h\}} \alpha_j v_j$, we infer that $0 = -\frac{1}{h-1} \sum_{j\in\{2,\ldots,h\}} \alpha_j$. Looking at a coordinate

where $v_i$ is 1, we infer that

$$0 = \alpha_i - \frac{1}{h-1} \sum_{j \in \{2,\ldots,h\} \setminus \{i\}} \alpha_j = \alpha_i \left(1 + \frac{1}{h-1}\right) - \frac{1}{h-1} \sum_{j \in \{2,\ldots,h\}} \alpha_j = \alpha_i \left(1 + \frac{1}{h-1}\right)$$

and hence $\alpha_i = 0$, $i \in \{2, \ldots, h\}$. Similarly, we can show that the vectors $v_{h+1}, \ldots, v_{h+k-1}$ as well as the vectors $v_{h+k}, \ldots, v_{hk}$ are linearly independent.

Since we assume $a > b > c > d \geq 0$, we clearly have

$$\big(a + (h-1)c\big) + (k-1)\big(b + (h-1)d\big) > \big(a - c\big) + (k-1)\big(b - d\big) > 0$$

and

$$\begin{aligned} &\big(a + (h-1)c\big) + (k-1)\big(b + (h-1)d\big) > \\ &\big(a + (h-1)c\big) - \big(b + (h-1)d\big) = (a-b) + (h-1)(c-d) > 0, \end{aligned}$$

which shows that $\lambda_1 > \lambda_2 = \ldots = \lambda_h > 0$ and $\lambda_1 > \lambda_{h+1} = \ldots = \lambda_{h+k-1} > 0$. It is

$$\big(a - c\big) + (k-1)\big(b - d\big) > (a-c) - (b-d)$$

and

$$\big(a - c\big) + (k-1)\big(b - d\big) > -(a-c) + (b-d),$$

and also

$$\begin{aligned} \big(a + (h-1)c\big) - \big(b + (h-1)d\big) &= (a-b) + (h-1)(c-d) \\ &\geq (a-b) + (c-d) > (a-b) + (d-c) \\ &= (a-c) - (b-d) \end{aligned}$$

and

$$\begin{aligned}\big(a+(h-1)c\big)-\big(b+(h-1)d\big) &= (a-b)+(h-1)(c-d)\\ &\geq (a-b)+(c-d) > (b-a)+(c-d)\\ &= -(a-c)+(b-d),\end{aligned}$$

which shows $\lambda_2 = \ldots = \lambda_h > |\lambda_{h+k}| = \ldots = |\lambda_{hk}|$ and $\lambda_{h+1} = \ldots = \lambda_{h+k-1} > |\lambda_{h+k}| = \ldots = |\lambda_{hk}|$. □

Note that we have

$$f^{(s)} - \frac{|V_s|}{n}\cdot \mathbf{1}_n = f^{(s)} - \frac{1}{h}\cdot \mathbf{1}_n = \frac{h-1}{h}v_{1+s}, \quad s \in [h-1], \tag{5.27}$$

where $f^{(s)}$ is the group-membership vector of $V_s$ and $f^{(s)} - \frac{|V_s|}{n}$ is the vector encountered in the second step of Algorithm 12 or Algorithm 13.

The next lemma provides an orthonormal basis of the eigenspace associated with eigenvalue $\lambda_{h+1} = \ldots = \lambda_{h+k-1}$.

**Lemma 5.9.2.** *An orthonormal basis of the eigenspace of $\widetilde{\mathcal{W}}$ corresponding to the eigenvalue $\lambda_{h+1} = \ldots = \lambda_{h+k-1}$ is given by $\{n_1, \ldots, n_{k-1}\}$ with*

$$n_1 = \begin{pmatrix}[(k-1)q_1]\\ [-q_1]\\ [-q_1]\\ [-q_1]\\ \vdots\\ [-q_1]\\ [-q_1]\end{pmatrix} \; n_2 = \begin{pmatrix}[0]\\ [(k-2)q_2]\\ [-q_2]\\ [-q_2]\\ \vdots\\ [-q_2]\\ [-q_2]\end{pmatrix} \; \ldots n_i = \begin{pmatrix}[0]\\ \vdots\\ [0]\\ [(k-i)q_i]\\ [-q_i]\\ \vdots\\ [-q_i]\end{pmatrix} \; \ldots n_{k-1} = \begin{pmatrix}[0]\\ [0]\\ \vdots\\ [0]\\ [0]\\ [q_{k-1}]\\ [-q_{k-1}]\end{pmatrix} \tag{5.28}$$

*where for $z \in \mathbb{R}$, by $[z]$ we denote a block of size $\frac{n}{k}$ with all entries equaling $z$ and*

$$q_i = \frac{1}{\sqrt{\left(\frac{n}{k}(k-i)^2 + \frac{n}{k}(k-i)\right)}}, \quad i \in [k-1]. \tag{5.29}$$

*Proof.* It is easy to verify that any $n_i$ is indeed an eigenvector of $\widetilde{\mathcal{W}}$ with eigenvalue $\lambda_{h+1} = \ldots = \lambda_{h+k-1}$, $i \in [k-1]$. Furthermore, we have

$$\|n_i\|^2 = q_i^2 \left(\frac{n}{k}(k-i)^2 + \frac{n}{k}(k-i)\right) = 1, \quad i \in [k-1],$$

and

$$\langle n_i, n_j \rangle = \frac{n}{k}\big( - q_i \cdot (k-j) q_j\big) + \frac{n}{k}(k-j)(q_i \cdot q_j) = 0, \quad 1 \le i < j \le n.$$

□

Let $\mathcal{L}$ be the expected Laplacian matrix. We have $\mathcal{L} = \mathcal{D} - \mathcal{W}$, where $\mathcal{D}$ is the expected degree matrix. The expected degree of vertex $i$ in a random graph constructed according to our variant of the stochastic block model equals $\sum_{j \in [n] \setminus \{i\}} \mathcal{W}_{ij} = \lambda_1 - a$ (with $\lambda_1$ defined in (5.26)) and hence $\mathcal{D} = (\lambda_1 - a) I_n$.

The following lemma characterizes the eigenvalues and eigenvectors of $Z^T \mathcal{L} Z$, where $Z \in \mathbb{R}^{n \times (n-h+1)}$ is the matrix computed in the execution of Algorithm 12 or Algorithm 13.

**Lemma 5.9.3.** *Let $Z \in \mathbb{R}^{n \times (n-h+1)}$ be any matrix whose columns form an orthonormal basis of the nullspace of $F^T$, where $F$ is the matrix that has the vectors $f^{(s)} - \frac{|V_s|}{n} \cdot \mathbf{1}_n$, $s \in [h-1]$, as columns. Then the eigenvalues of $Z^T \mathcal{L} Z$ are*

$$\lambda_1 - \lambda_1, \lambda_1 - \lambda_{h+1}, \lambda_1 - \lambda_{h+2}, \ldots, \lambda_1 - \lambda_n$$

*with $\lambda_i$ defined in* (5.26)*. It is*

$$\begin{aligned}
&\lambda_1 - \lambda_1 = 0,\\
&\lambda_1 - \lambda_{h+1} = \lambda_1 - \lambda_{h+2} = \ldots = \lambda_1 - \lambda_{h+k-1},\\
&\lambda_1 - \lambda_{h+k} = \lambda_1 - \lambda_{h+k+1} = \ldots = \lambda_1 - \lambda_{hk},\\
&\lambda_1 - \lambda_{hk+1} = \lambda_1 - \lambda_{hk+2} = \ldots = \lambda_1 - \lambda_n = \lambda_1
\end{aligned} \tag{5.30}$$

*with*

$$\lambda_1 - \lambda_1 < \lambda_1 - \lambda_{h+1} < \min\{\lambda_1 - \lambda_{h+k}, \lambda_1 - \lambda_{hk+1}\}, \tag{5.31}$$

*so that the $k$ smallest eigenvalues of $Z^T\mathcal{L}Z$ are $\lambda_1 - \lambda_1, \lambda_1 - \lambda_{h+1}, \lambda_1 - \lambda_{h+2}, \ldots, \lambda_1 - \lambda_{h+k-1}$.*

*Furthermore, there exists an orthonormal basis $\{r_1, r_{h+1}, r_{h+2}, \ldots, r_n\}$ of eigenvectors of $Z^T\mathcal{L}Z$ with $r_i$ corresponding to eigenvalue $\lambda_1 - \lambda_i$ such that*

$$Zr_1 = \mathbf{1}_n/\sqrt{n} \qquad \textit{and} \qquad Zr_{h+i} = n_i, \quad i \in [k-1],$$

*with $n_i$ defined in* (5.28)*.*

*Proof.* Because of $Z^TZ = I_{(n-h+1)}$ we have

$$\begin{aligned}
Z^T\mathcal{L}Z = Z^T(\mathcal{D} - \mathcal{W})Z = Z^T\mathcal{D}Z - Z^T(\widetilde{\mathcal{W}} - aI_n)Z = (\lambda_1 - a)I_n - Z^T\widetilde{\mathcal{W}}Z + aI_n\\
= \lambda_1 I_n - Z^T\widetilde{\mathcal{W}}Z.
\end{aligned}$$

Let $\{u_1, \ldots, u_n\}$ be an orthonormal basis of eigenvectors of $\widetilde{\mathcal{W}}$ with $u_i$ corresponding to eigenvalue $\lambda_i$. According to Lemma 5.9.1 and Lemma 5.9.2 we can choose $u_1 = \mathbf{1}_n/\sqrt{n}$ and $u_{h+i} = n_i$ for $i \in [k-1]$. We can write $\widetilde{\mathcal{W}}$ as $\widetilde{\mathcal{W}} = \sum_{i=1}^n \lambda_i u_i u_i^T$.

The nullspace of $F^T$, where $F$ is the matrix that has the vectors $f^{(s)} - \frac{|V_s|}{n}\cdot\mathbf{1}_n$, $s \in [h-1]$,

as columns, equals the orthogonal complement of $\{f^{(s)} - (|V_s|/n) \cdot \mathbf{1}_n, s \in [h-1]\}$. According to (5.27), the orthogonal complement of $\{f^{(s)} - (|V_s|/n) \cdot \mathbf{1}_n, s \in [h-1]\}$ equals the orthogonal complement of $\{v_{1+s}, s \in [h-1]\}$, with $v_i$ defined in Lemma 5.9.1 and being an eigenvalue of $\widetilde{\mathcal{W}}$ with eigenvalue $\lambda_i$. According to Lemma 5.9.1, $\{v_{1+s}, s \in [h-1]\}$ is a basis of the eigenspace of $\widetilde{\mathcal{W}}$ corresponding to eigenvalue $\lambda_2 = \lambda_3 = \ldots = \lambda_h$, and hence the orthogonal complement of $\{v_{1+s}, s \in [h-1]\}$ equals the orthogonal complement of $\{u_2, \ldots, u_h\}$, which is the subspace spanned by $\{u_1, u_{h+1}, u_{h+2}, \ldots, u_n\}$. Let $U \in \mathbb{R}^{n \times (n-h+1)}$ be a matrix that has the vectors $u_1, u_{h+1}, u_{h+2}, \ldots, u_n$ as columns (in this order). It follows that $U = ZR$ for some $R \in \mathbb{R}^{(n-h+1)\times(n-h+1)}$ with $R^T R = RR^T = I_{(n-h+1)}$. It is

$$Z^T \mathcal{L} Z = \lambda_1 I_n - Z^T \widetilde{\mathcal{W}} Z = \lambda_1 I_n - RU^T \left( \sum_{i=1}^{n} \lambda_i u_i u_i^T \right) UR^T.$$

Let $r_1$ be the first column of $R$, $r_{h+1}$ be the second column of $R$, $r_{h+2}$ be the third column of $R$, and so on. We have

$$\begin{aligned}
Z^T \mathcal{L} Z r_1 &= \left[ \lambda_1 I_n - RU^T \left( \sum_{i=1}^{n} \lambda_i u_i u_i^T \right) UR^T \right] r_1 \\
&= \lambda_1 r_1 - RU^T \left( \sum_{i=1}^{n} \lambda_i u_i u_i^T \right) U e_1 \\
&= \lambda_1 r_1 - RU^T \left( \sum_{i=1}^{n} \lambda_i u_i u_i^T \right) u_1 \\
&= \lambda_1 r_1 - \lambda_1 RU^T u_1 \\
&= \lambda_1 r_1 - \lambda_1 R e_1 \\
&= (\lambda_1 - \lambda_1) r_1,
\end{aligned}$$

where $e_1$ denotes the first natural basis vector. Similarly, we obtain $Z^T \mathcal{L} Z r_{h+i} = (\lambda_1 - \lambda_{h+i}) r_{h+i}$, $i \in [n-h]$. This proves that the eigenvalues of $Z^T \mathcal{L} Z$ are $\lambda_1 - \lambda_1, \lambda_1 - \lambda_{h+1}, \lambda_1 - \lambda_{h+2}, \ldots, \lambda_1 - \lambda_n$. The claims in (5.30) and (5.31) immediately follow from Lemma 5.9.1.

Clearly, it is $Zr_1 = u_1 = \mathbf{1}_n/\sqrt{n}$ and $Zr_{h+i} = u_{h+i} = n_i$ for $i \in [k-1]$. □

We need one more simple lemma.

**Lemma 5.9.4.** *Let $T \in \mathbb{R}^{n\times k}$ be a matrix that contains the vectors $\mathbf{1}_n/\sqrt{n}, n_1, n_2, \ldots, n_{k-1}$, with $n_i$ defined in* (5.28)*, as columns. For $i \in [n]$, let $t_i$ denote the $i$-th row of $T$. For all $i, j \in [n]$, we have $t_i = t_j$ if and only if the vertices $i$ and $j$ are in the same cluster $C_l$. If the vertices $i$ and $j$ are not in the same cluster, then $\|t_i - t_j\| = \sqrt{2k/n}$.*

*Proof.* This simply follows from the structure of the vectors $n_i$. It is, up to a permutation of the entries,

$$t_i = \left(\frac{1}{\sqrt{n}}, -q_1, -q_2, \ldots, -q_{l-1}, (k-l)q_l, 0, 0, \ldots, 0\right),$$

with $q_l$ defined in (5.29), for all $i \in [n]$ such that vertex $i$ is in cluster $C_l$, $l \in [k-1]$, and

$$t_i = \left(\frac{1}{\sqrt{n}}, -q_1, -q_2, \ldots, -q_{k-1}\right)$$

for all $i \in [n]$ such that vertex $i$ is in cluster $C_k$. It is easy to verify that $\|t_i - t_j\|^2 = 2k/n$ for all $i, j \in [n]$ such that the vertices $i$ and $j$ are not in the same cluster. □

**Part 2: Deviation of $Z^T LZ$ from $Z^T \mathcal{L} Z$**

We want to obtain an upper bound on $\|Z^T LZ - Z^T \mathcal{L} Z\|$. Because of $Z^T Z = I_{(n-h+1)}$, it is $\|Z\| = \|Z^T\| = 1$ and hence

$$\|Z^T LZ - Z^T \mathcal{L} Z\| \leq \|Z^T\| \cdot \|L - \mathcal{L}\| \cdot \|Z\| \leq \|L - \mathcal{L}\|. \tag{5.32}$$

We have

$$\|L - \mathcal{L}\| = \|(D - W) - (\mathcal{D} - \mathcal{W})\| \leq \|D - \mathcal{D}\| + \|W - \mathcal{W}\|,$$

with $\mathcal{D} = (\lambda_1 - a)I_n$ as we have seen in Part 1. We bound both terms separately.

- Upper bound on $\|W - \mathcal{W}\|$:

  Theorem 5.2 of [110] provides a bound on $\|W - \mathcal{W}\|$: assuming that $a \geq C \ln n/n$ for some $C > 0$, for every $r > 0$ there exists a constant $\mathrm{const}(C, r)$ such that

  $$\|W - \mathcal{W}\| \leq \mathrm{const}(C, r)\sqrt{a \cdot n} \tag{5.33}$$

  with probability at least $1 - n^{-r}$.

- Upper bound on $\|D - \mathcal{D}\|$:

  The matrix $D - \mathcal{D}$ is a diagonal matrix and hence $\|D - \mathcal{D}\| = \max_{i\in[n]} |D_{ii} - \mathcal{D}_{ii}| = \max_{i\in[n]} |D_{ii} - (\lambda_1 - a)|$. The random variable $D_{ii} = \sum_{j\in[n]\setminus\{i\}} \mathbb{1}[i \sim j]$, where $\mathbb{1}[i \sim j]$ denotes the indicator function of the event that there is an edge between vertices $i$ and $j$, is a sum of independent Bernoulli random variables. It is $\mathbb{E}[D_{ii}] = \lambda_1 - a$. For a fixed $i \in [n]$, we want to obtain an upper bound on $|D_{ii} - (\lambda_1 - a)| = |D_{ii} - \mathbb{E}[D_{ii}]|$ and distinguish two cases:

  1. $a > \frac{1}{2}$:

     Hoeffding's inequality [e.g., 138, Theorem 1] yields

     $$\Pr[|D_{ii} - (\lambda_1 - a)| \geq t] \leq 2\exp\left(-\frac{2t^2}{n}\right)$$

     for any $t > 0$. Choosing $t = \sqrt{2(r+1)}\sqrt{a \cdot n \ln n}$ for $r > 0$, we have with $\mathrm{const}(r) = \sqrt{2(r+1)}$ that

     $$\Pr\left[|D_{ii} - (\lambda_1 - a)| \geq \mathrm{const}(r) \cdot \sqrt{a \cdot n \ln n}\right] \leq \tag{5.34}$$
     $$2\exp\left(-4(r+1)a \ln n\right) \leq n^{-(r+1)}.$$

2. $a \leq \frac{1}{2}$:

   Bernstein's inequality [e.g., 138, Theorem 3] yields

$$\Pr\left[|D_{ii} - (\lambda_1 - a)| > tn\right] \leq 2\exp\left(-\frac{nt^2}{2\left(\frac{\mathrm{Var}[D_{ii}]}{n} + \frac{t}{3}\right)}\right)$$

   for any $t > 0$. It is

$$\mathrm{Var}[D_{ii}] = \sum_{j \in [n]\setminus\{i\}} \mathrm{Var}\left[\mathbb{1}[i \sim j]\right] =$$
$$\sum_{j \in [n]\setminus\{i\}} \Pr[\mathbb{1}[i \sim j]](1 - \Pr[\mathbb{1}[i \sim j]]) \leq na(1-a) \leq na$$

   since the function $x \mapsto x(1-x)$ is monotonically increasing on $[0, 1/2]$. If we choose $t = \mathrm{const} \cdot \frac{\sqrt{a \cdot n \ln n}}{n}$ for some $\mathrm{const} > 0$, assuming that $a \geq C \ln n / n$ for some $C > 0$, we have

$$\frac{\mathrm{Var}[D_{ii}]}{n} + \frac{t}{3} \leq a\left(1 + \frac{\mathrm{const}}{3\sqrt{C}}\right)$$

   and hence

$$\Pr\left[|D_{ii} - (\lambda_1 - a)| > \mathrm{const} \cdot \sqrt{a \cdot n \ln n}\right] \leq 2\exp\left(-\frac{\mathrm{const}^2 \cdot \ln n}{2 + \frac{2\,\mathrm{const}}{3\sqrt{C}}}\right).$$

   Because of

$$\frac{\mathrm{const}^2}{2 + \frac{2\,\mathrm{const}}{3\sqrt{C}}} \to \infty \quad \text{as const} \to \infty,$$

   for every $r > 0$ we can choose $\mathrm{const} = \mathrm{const}(C, r)$ large enough such that

$\text{const}^2 / \left(2 + \frac{2\,\text{const}}{3\sqrt{C}}\right) \geq 2(r+1)$ and

$$\Pr\left[|D_{ii} - (\lambda_1 - a)| > \text{const}(C,r) \cdot \sqrt{a \cdot n \ln n}\right] \leq n^{-(r+1)}. \tag{5.35}$$

Choosing $\text{const}(C,r)$ as the maximum of $\text{const}(r)$ encountered in (5.34) and $\text{const}(C,r)$ encountered in (5.35), we see that there exists $\text{const}(C,r)$ such that

$$\Pr\left[|D_{ii} - (\lambda_1 - a)| > \text{const}(C,r) \cdot \sqrt{a \cdot n \ln n}\right] \leq n^{-(r+1)},$$

no matter whether $a > 1/2$ or $1/2 \geq a \geq C \ln n/n$. Applying a union bound we obtain

$$\Pr\left[\max_{i\in[n]} |D_{ii} - (\lambda_1 - a)| > \text{const}(C,r) \cdot \sqrt{a \cdot n \ln n}\right] \leq n \cdot n^{-(r+1)} = n^{-r},$$

and hence with probability at least $1 - n^{-r}$ we have

$$\|D - \mathcal{D}\| \leq \text{const}(C,r)\sqrt{a \cdot n \ln n}. \tag{5.36}$$

From (5.33) and (5.36) we see that for every $r > 0$ there exists $\text{const}(C,r)$ such that with probability at least $1 - n^{-r}$ we have

$$\|W - \mathcal{W}\| \leq \text{const}(C,r)\sqrt{a \cdot n} \qquad \text{and} \qquad \|D - \mathcal{D}\| \leq \text{const}(C,r)\sqrt{a \cdot n \ln n} \tag{5.37}$$

and hence

$$\|Z^T L Z - Z^T \mathcal{L} Z\| \leq \|L - \mathcal{L}\| \leq \|D - \mathcal{D}\| + \|W - \mathcal{W}\| \leq \text{const}(C,r)\sqrt{a \cdot n \ln n}. \tag{5.38}$$

For illustrative purposes, we show empirically that, in general, our upper bounds on

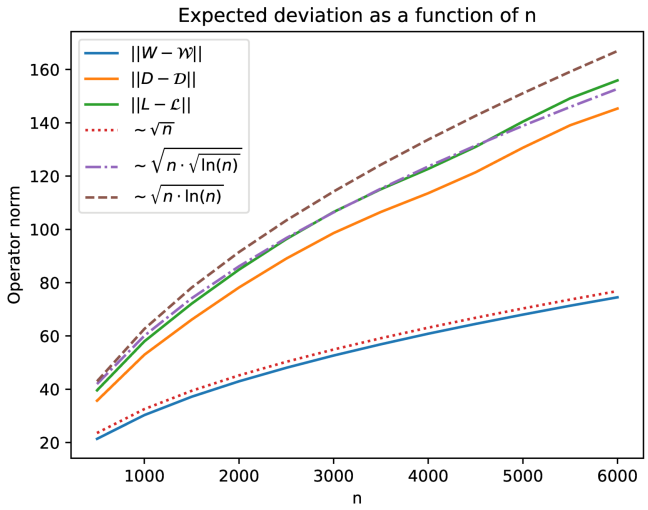



Figure 5.7: Average deviations $\|W - \mathcal{W}\|$, $\|D - \mathcal{D}\|$ and $\|L - \mathcal{L}\|$ as a function of $n$ when $a = 0.6, b = 0.5, c = 0.4, d = 0.3$ are constant, $k = 5$ and $h = 2$. The average is computed over sampling the graph for 100 times.

$\|W - \mathcal{W}\|$, $\|D - \mathcal{D}\|$ and $\|L - \mathcal{L}\|$ in (5.37) and (5.38), respectively, are tight, up to a factor of at most $\sqrt[4]{\ln n}$ in case of $\|D - \mathcal{D}\|$ and $\|L - \mathcal{L}\|$. The plot in Figure 5.7 shows the observed deviations $\|W - \mathcal{W}\|$, $\|D - \mathcal{D}\|$ and $\|L - \mathcal{L}\|$ as a function of $n$ when $a = 0.6, b = 0.5, c = 0.4, d = 0.3$ are constant, $k = 5$ and $h = 2$. The shown curves are average results, obtained from sampling the graph for 100 times.

**Part 3: Proving Theorem 5.4.1 for Algorithm 12 (unnormalized SC with fairness constraints)**

In the last step of Algorithm 12 we apply $k$-means clustering to the rows of the matrix $ZY$, where $Y \in \mathbb{R}^{(n-h+1)\times k}$ contains some orthonormal eigenvectors corresponding to the $k$ smallest eigenvalues of $Z^T L Z$ as columns. We want to show that up to some orthogonal transformation, the rows of $ZY$ are close to the rows of $Z\mathcal{Y}$, where $\mathcal{Y} \in \mathbb{R}^{(n-h+1)\times k}$ contains some orthonormal eigenvectors corresponding to the $k$ smallest eigenvalues of $Z^T \mathcal{L} Z$ as columns. According to Lemma 5.9.3, we can choose $\mathcal{Y}$ in such a way that $Z\mathcal{Y} = T$ with $T$ as in Lemma 5.9.4, that is $T$ contains the vectors $\mathbf{1}_n/\sqrt{n}, n_1, n_2, \ldots, n_{k-1}$, with $n_i$ defined in (5.28), as columns.

We want to obtain an upper bound on $\min_{U\in\mathbb{R}^{k\times k}:U^TU=UU^T=I_k}\|Z\mathcal{Y}-ZYU\|_F$. For any $U\in\mathbb{R}^{k\times k}$ with $U^TU=UU^T=I_k$, because of $Z^TZ=I_{(n-h+1)}$ we have

$$\|Z\mathcal{Y}-ZYU\|_F^2=\|Z(\mathcal{Y}-YU)\|_F^2=\mathrm{Tr}((\mathcal{Y}-YU)^TZ^TZ(\mathcal{Y}-YU))=\|\mathcal{Y}-YU\|_F^2$$

and hence

$$\min_{U\in\mathbb{R}^{k\times k}:U^TU=UU^T=I_k}\|Z\mathcal{Y}-ZYU\|_F=\min_{U\in\mathbb{R}^{k\times k}:U^TU=UU^T=I_k}\|\mathcal{Y}-YU\|_F. \tag{5.39}$$

We proceed similarly to [110]. According to Proposition 2.2 and Equation (2.6) in [139] we have (note that the set of all orthogonal matrices $U\in\mathbb{R}^{k\times k}$ is a compact subset of $\mathbb{R}^{k\times k}$ and hence the infimum is indeed a minimum)

$$\begin{aligned}&\min_{U\in\mathbb{R}^{k\times k}:U^TU=UU^T=I_k}\|\mathcal{Y}-YU\|_F\leq \\ &\sqrt{2}\|\mathcal{Y}\mathcal{Y}^T(I_{(n-h+1)}-YY^T)\|_F\overset{(5.23)}{\leq}\sqrt{2}\sqrt{k}\,\|\mathcal{Y}\mathcal{Y}^T(I_{(n-h+1)}-YY^T)\|.\end{aligned} \tag{5.40}$$

According to Lemma 5.9.3 the eigenvalues of $Z^T\mathcal{L}Z$ are $\lambda_1-\lambda_1,\lambda_1-\lambda_{h+1},\lambda_1-\lambda_{h+2},\ldots,\lambda_1-\lambda_n$. The $k$ smallest eigenvalues are $\lambda_1-\lambda_1,\lambda_1-\lambda_{h+1},\lambda_1-\lambda_{h+2},\ldots,\lambda_1-\lambda_{h+k-1}$ and the $(k+1)$-th smallest eigenvalue is either $\lambda_1-\lambda_{h+k}$ or $\lambda_1$. Hence, for the eigengap $\gamma$ between the $k$-th and the $(k+1)$-th smallest eigenvalue we have

$$\begin{aligned}\gamma&=\min\{(\lambda_1-\lambda_{h+k})-(\lambda_1-\lambda_{h+k-1}),\lambda_1-(\lambda_1-\lambda_{h+k-1})\}\\&=\min\left\{\frac{n}{k}(c-d),\frac{n}{kh}((a-b)+(h-1)(c-d))\right\}.\end{aligned}$$

It is

$$\frac{n}{2k}(c-d)\leq\frac{n(h-1)}{hk}(c-d)\leq\gamma\leq\frac{n}{k}(c-d). \tag{5.41}$$

We want to show that

$$\|\mathcal{Y}\mathcal{Y}^T(I_{(n-h+1)} - YY^T)\| \leq \frac{4}{\gamma}\|Z^T\mathcal{L}Z - Z^TLZ\|. \tag{5.42}$$

If $\|Z^T\mathcal{L}Z - Z^TLZ\| > \frac{\gamma}{4}$, then (5.42) holds trivially because of

$$\|\mathcal{Y}\mathcal{Y}^T(I_{(n-h+1)} - YY^T)\| \leq \|\mathcal{Y}\mathcal{Y}^T\| \cdot \|I_{(n-h+1)} - YY^T\| = 1 \cdot 1 = 1.$$

Assume that $\|Z^T\mathcal{L}Z - Z^TLZ\| \leq \frac{\gamma}{4}$ and let $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_{n-h+1}$ be the eigenvalues of $Z^TLZ$. Since $L$ is positive semi-definite, so is $Z^TLZ$, and hence $\mu_1 \geq 0$. Let $\lambda'_1 \leq \lambda'_2 \leq \ldots, \lambda'_{n-h+1}$ be the eigenvalues $\lambda_1 - \lambda_1, \lambda_1 - \lambda_{h+1}, \lambda_1 - \lambda_{h+2}, \ldots, \lambda_1 - \lambda_n$ of $Z^T\mathcal{L}Z$ in ascending order. According to Weyl's Perturbation Theorem [e.g., 140, Corollary III.2.6] it is

$$|\mu_i - \lambda'_i| \leq \|Z^T\mathcal{L}Z - Z^TLZ\| \leq \frac{\gamma}{4}, \quad i \in [n-h+1].$$

In particular, we have

$$\mu_1, \ldots, \mu_k \in \left[0, \lambda'_k + \frac{\gamma}{4}\right], \quad \mu_{k+1}, \ldots, \mu_n \in \left[\lambda'_{k+1} - \frac{\gamma}{4}, \infty\right)$$

with $\left(\lambda'_{k+1} - \frac{\gamma}{4}\right) - \left(\lambda'_k + \frac{\gamma}{4}\right) = \frac{\gamma}{2}$. The Davis-Kahan sin$\Theta$ Theorem [e.g., 140, Theorem VII.3.1] yields that

$$\|\mathcal{Y}\mathcal{Y}^T(I_{(n-h+1)} - YY^T)\| \leq \frac{2}{\gamma}\|Z^T\mathcal{L}Z - Z^TLZ\|$$

and hence (5.42). Combining (5.39) to (5.42), we end up with

$$\min_{U \in \mathbb{R}^{k\times k}: U^TU = UU^T = I_k} \|Z\mathcal{Y} - ZYU\|_F \leq \frac{16\sqrt{k^3}}{n(c-d)}\|Z^T\mathcal{L}Z - Z^TLZ\|. \tag{5.43}$$

Using (5.38) from Part 2, we see that with probability at least $1 - n^{-r}$

$$\min_{U \in \mathbb{R}^{k \times k} : U^T U = U U^T = I_k} \|Z\mathcal{Y} - ZYU\|_F \leq \text{const}(C, r) \cdot \frac{\sqrt{k^3}}{c - d} \cdot \sqrt{\frac{a \cdot \ln n}{n}}. \tag{5.44}$$

We use Lemma 5.3 in [110] to complete the proof of Theorem 5.4.1 for Algorithm 12. Assume that (5.44) holds and let $U \in \mathbb{R}^{k \times k}$ be an orthogonal matrix attaining the minimum, that is we have

$$\|Z\mathcal{Y}U^T - ZY\|_F = \|Z\mathcal{Y} - ZYU\|_F \leq \text{const}(C, r) \cdot \frac{\sqrt{k^3}}{c - d} \cdot \sqrt{\frac{a \cdot \ln n}{n}}. \tag{5.45}$$

As we have noted above, we can choose $\mathcal{Y}$ in such a way that $Z\mathcal{Y} = T$ with $T$ as in Lemma 5.9.4. According to Lemma 5.9.4, if we denote the $i$-th row of $T$ by $t_i$, then $t_i = t_j$ if the vertices $i$ and $j$ are in the same cluster and $\|t_i - t_j\| = \sqrt{2k/n}$ if the vertices $i$ and $j$ are not in the same cluster. Since multiplying $T$ by $U^T$ from the right side has the effect of applying an orthogonal transformation to the rows of $T$, the same properties are true for the matrix $TU^T$. Lemma 5.3 in [110] guarantees that for any $\delta \leq \sqrt{2k/n}$, if

$$\frac{16 + 8M}{\delta^2} \|TU^T - ZY\|_F^2 < \underbrace{|C_l|}_{= \frac{n}{k}}, \quad l \in [k], \tag{5.46}$$

with $|C_l|$ being the size of cluster $C_l$, then a $(1 + M)$-approximation algorithm for $k$-means clustering applied to the rows of the matrix $ZY$ returns a clustering that misclassifies at most

$$\frac{4(4 + 2M)}{\delta^2} \|TU^T - ZY\|_F^2 \tag{5.47}$$

many vertices. If we choose $\delta = \sqrt{2k/n}$, then for a small enough $\widehat{C}_1 = \widehat{C}_1(C, r)$, because of (5.45), the condition (5.11) implies (5.46). Also, for a large enough $\widetilde{C}_1 = \widetilde{C}_1(C, r)$, the expression (5.47) is upper bounded by the expression (5.12).

**Part 4: Proving Theorem 5.4.1 for Algorithm 13 (normalized SC with fairness constraints)**

According to Part 2, for every $r > 0$ there exists $\mathrm{const}(C, r)$ such that with probability at least $1 - n^{-r}$ we have

$$\|D - \mathcal{D}\| \leq \mathrm{const}(C, r)\sqrt{a \cdot n \ln n}. \tag{5.48}$$

Condition (5.13), with a suitable $\widehat{C}_2 = \widehat{C}_2(C, r)$, implies that in this case we also have

$$\|D - \mathcal{D}\| \leq \frac{\lambda_1 - a}{2}. \tag{5.49}$$

Let $\mu'_1, \ldots, \mu'_{n-h+1}$ denote the eigenvalues of $Z^T D Z$. It is $\mathcal{D} = (\lambda_1 - a)I_n$ (see Part 1) and because of $Z^T Z = I_{(n-h+1)}$ we have $Z^T \mathcal{D} Z = (\lambda_1 - a)I_{(n-h+1)}$. According to Weyl's Perturbation Theorem [e.g., 140, Corollary III.2.6] it is

$$|\mu'_i - (\lambda_1 - a)| \leq \|Z^T D Z - Z^T \mathcal{D} Z\| \leq \|D - \mathcal{D}\|, \quad i \in [n - h + 1], \tag{5.50}$$

where the second inequality follows analogously to (5.32). It follows from (5.49) that

$$\mu'_i \geq \frac{\lambda_1 - a}{2} \overset{(5.13)}{>} 0, \quad i \in [n - h + 1], \tag{5.51}$$

In particular, this shows that $Z^T D Z$ is positive definite and hence Algorithm 13 is well-defined.

Now we proceed similarly to Part 3. In the last step of Algorithm 13 we apply $k$-means clustering to the rows of the matrix $ZQ^{-1}X$, where $Q \in \mathbb{R}^{(n-h+1)\times(n-h+1)}$ is the positive definite square root of $Z^T D Z$ and $X \in \mathbb{R}^{(n-h+1)\times k}$ contains some orthonormal eigenvectors corresponding to the $k$ smallest eigenvalues of $Q^{-1}Z^T L Z Q^{-1}$ as columns. We want to show that up to some orthogonal transformation, the rows of $ZQ^{-1}X$ are

close to the rows of $Z\mathcal{Q}^{-1}\mathcal{X}$, where $\mathcal{Q} \in \mathbb{R}^{(n-h+1)\times(n-h+1)}$ is the positive definite square root of $Z^T\mathcal{D}Z$ and $\mathcal{X} \in \mathbb{R}^{(n-h+1)\times k}$ contains some orthonormal eigenvectors corresponding to the $k$ smallest eigenvalues of $\mathcal{Q}^{-1}Z^T\mathcal{L}Z\mathcal{Q}^{-1}$ as columns. It is $Z^T\mathcal{D}Z = (\lambda_1 - a)I_{(n-h+1)}$. Consequently, $\mathcal{Q} = \sqrt{\lambda_1 - a} \cdot I_{(n-h+1)}$ and $\mathcal{Q}^{-1} = \frac{1}{\sqrt{\lambda_1 - a}} \cdot I_{(n-h+1)}$ and it is $\mathcal{Q}^{-1}Z^T\mathcal{L}Z\mathcal{Q}^{-1} = \frac{1}{\lambda_1 - a} \cdot Z^T\mathcal{L}Z$. Hence, the eigenvalues of $\mathcal{Q}^{-1}Z^T\mathcal{L}Z\mathcal{Q}^{-1}$ are the eigenvalues of $Z^T\mathcal{L}Z$ rescaled by $(\lambda_1 - a)^{-1}$ with the same eigenvectors as for $Z^T\mathcal{L}Z$. According to Lemma 5.9.3, we can choose $\mathcal{X}$ in such a way that $Z\mathcal{Q}^{-1}\mathcal{X} = \frac{1}{\sqrt{\lambda_1 - a}} \cdot Z\mathcal{X} = \frac{1}{\sqrt{\lambda_1 - a}} \cdot T$ with $T$ as in Lemma 5.9.4, that is $T$ contains the vectors $\mathbf{1}_n/\sqrt{n}, n_1, n_2, \ldots, n_{k-1}$, with $n_i$ defined in (5.28), as columns.

We want to obtain an upper bound on $\min_{U \in \mathbb{R}^{k\times k}: U^TU = UU^T = I_k} \|Z\mathcal{Q}^{-1}\mathcal{X} - ZQ^{-1}XU\|_F$. Analogously to (5.39) we obtain

$$\begin{aligned} &\min_{U \in \mathbb{R}^{k\times k}: U^TU = UU^T = I_k} \|Z\mathcal{Q}^{-1}\mathcal{X} - ZQ^{-1}XU\|_F \\ &= \min_{U \in \mathbb{R}^{k\times k}: U^TU = UU^T = I_k} \|\mathcal{Q}^{-1}\mathcal{X} - Q^{-1}XU\|_F. \end{aligned}$$

The rank of both $\mathcal{Q}^{-1}\mathcal{X}$ and $Q^{-1}XU$ equals $k$ and hence the rank of $\mathcal{Q}^{-1}\mathcal{X} - Q^{-1}XU$ is not greater than $2k$. We have

$$\begin{aligned} \|\mathcal{Q}^{-1}\mathcal{X} - Q^{-1}XU\|_F &\overset{(5.23)}{\leq} \sqrt{2k} \cdot \|\mathcal{Q}^{-1}\mathcal{X} - Q^{-1}XU\| \\ &\leq \sqrt{2k} \cdot \|\mathcal{Q}^{-1}\| \cdot \|\mathcal{X} - XU\| + \sqrt{2k} \cdot \|\mathcal{Q}^{-1} - Q^{-1}\| \cdot \|XU\| \end{aligned}$$

with $\|\mathcal{Q}^{-1}\| = \frac{1}{\sqrt{\lambda_1 - a}}$ and $\|XU\| = 1$ because of $X^TX = I_k$ and $U^TU = I_k$. Hence

$$\begin{aligned} &\min_{U:\, U^TU = UU^T = I_k} \|Z\mathcal{Q}^{-1}\mathcal{X} - ZQ^{-1}XU\|_F \\ &\leq \frac{\sqrt{2k}}{\sqrt{\lambda_1 - a}} \cdot \min_{U:\, U^TU = UU^T = I_k} \|\mathcal{X} - XU\| + \sqrt{2k} \cdot \|\mathcal{Q}^{-1} - Q^{-1}\|. \end{aligned} \tag{5.52}$$

Because of (5.23) we have

$$\min_{U\in\mathbb{R}^{k\times k}:U^TU=UU^T=I_k}\|\mathcal{X}-XU\|\leq\min_{U\in\mathbb{R}^{k\times k}:U^TU=UU^T=I_k}\|\mathcal{X}-XU\|_F \tag{5.53}$$

and similarly to how we obtained the bound (5.43) in Part 2, we can show that

$$\min_{U\in\mathbb{R}^{k\times k}:U^TU=UU^T=I_k}\|\mathcal{X}-XU\|_F\leq\frac{16\sqrt{k^3}(\lambda_1-a)}{n(c-d)}\|\mathcal{Q}^{-1}Z^T\mathcal{L}Z\mathcal{Q}^{-1}-Q^{-1}Z^TLZQ^{-1}\|. \tag{5.54}$$

Before looking at $\|\mathcal{Q}^{-1}Z^T\mathcal{L}Z\mathcal{Q}^{-1}-Q^{-1}Z^TLZQ^{-1}\|$ let us first look at the second term in (5.52). Because $Q^{-1}$ is symmetric and $\mathcal{Q}^{-1}=\frac{1}{\sqrt{\lambda_1-a}}\cdot I_{(n-h+1)}$ we have

$$\|\mathcal{Q}^{-1}-Q^{-1}\|=\max\left\{\left|\nu_i-\frac{1}{\sqrt{\lambda_1-a}}\right|:\nu_i\text{ is an eigenvalue of }Q^{-1}\right\}.$$

It is $Q^2=Z^TDZ$. Denoting the eigenvalues of $Z^TDZ$ by $\mu_1',\ldots,\mu_{n-h+1}'$ (note that all of them are greater than zero according to (5.51)), the eigenvalues of $Q^{-1}$ are $1/\sqrt{\mu_1'},\ldots,$ $1/\sqrt{\mu_{n-h+1}'}$. For any $z_1,z_2>0$ we have

$$|\sqrt{z_1}-\sqrt{z_2}|=\frac{|(\sqrt{z_1}-\sqrt{z_2})(\sqrt{z_1}+\sqrt{z_2})|}{\sqrt{z_1}+\sqrt{z_2}}=\frac{|z_1-z_2|}{\sqrt{z_1}+\sqrt{z_2}}\leq\frac{|z_1-z_2|}{\sqrt{z_2}} \tag{5.55}$$

and

$$\left|\frac{1}{\sqrt{z_1}}-\frac{1}{\sqrt{z_2}}\right|=\frac{|\sqrt{z_1}-\sqrt{z_2}|}{\sqrt{z_1}\sqrt{z_2}}\overset{\text{for }z_1\geq\frac{z_2}{2}}{\leq}\frac{\sqrt{2}\cdot|\sqrt{z_1}-\sqrt{z_2}|}{z_2}\overset{(5.55)}{\leq}\frac{\sqrt{2}\cdot|z_1-z_2|}{\sqrt{z_2^3}}. \tag{5.56}$$

According to (5.51) we have $\mu_i'\geq\frac{\lambda_1-a}{2}>0$, $i\in[n-h+1]$, and hence

$$\left|\frac{1}{\sqrt{\mu_i'}}-\frac{1}{\sqrt{\lambda_1-a}}\right|\overset{(5.56)}{\leq}\frac{\sqrt{2}\cdot|\mu_i'-(\lambda_1-a)|}{\sqrt{(\lambda_1-a)^3}}\overset{(5.50)}{\leq}\frac{\sqrt{2}\cdot\|D-\mathcal{D}\|}{\sqrt{(\lambda_1-a)^3}},\quad i\in[n-h+1],$$

and

$$\|\mathcal{Q}^{-1} - Q^{-1}\| \leq \frac{\sqrt{2} \cdot \|D - \mathcal{D}\|}{\sqrt{(\lambda_1 - a)^3}}. \tag{5.57}$$

Let us now look at $\|\mathcal{Q}^{-1} Z^T \mathcal{L} Z \mathcal{Q}^{-1} - Q^{-1} Z^T L Z Q^{-1}\|$. It is

$$\begin{aligned}
&\|\mathcal{Q}^{-1} Z^T \mathcal{L} Z \mathcal{Q}^{-1} - Q^{-1} Z^T L Z Q^{-1}\| \\
&\leq \|\mathcal{Q}^{-1} - Q^{-1}\| \cdot \|Z^T \mathcal{L} Z\| \cdot \|\mathcal{Q}^{-1}\| \\
&+ \|Q^{-1}\| \cdot \|Z^T \mathcal{L} Z - Z^T L Z\| \cdot \|\mathcal{Q}^{-1}\| + \|Q^{-1}\| \cdot \|Z^T L Z\| \cdot \|\mathcal{Q}^{-1} - Q^{-1}\|.
\end{aligned} \tag{5.58}$$

It is $\|\mathcal{Q}^{-1}\| = \frac{1}{\sqrt{\lambda_1 - a}}$. According to Lemma 5.9.3, the largest eigenvalue of $Z^T \mathcal{L} Z$ is $\lambda_1$ or $\lambda_1 - \lambda_{hk}$, where $\lambda_1 - \lambda_{hk} \leq 2\lambda_1$ according to Lemma 5.9.1. Consequently, $\|Z^T \mathcal{L} Z\| \leq 2\lambda_1$. It is

$$\|Q^{-1}\| \leq \|\mathcal{Q}^{-1} - Q^{-1}\| + \|\mathcal{Q}^{-1}\| \overset{(5.57)}{\leq} \frac{\sqrt{2} \cdot \|D - \mathcal{D}\|}{\sqrt{(\lambda_1 - a)^3}} + \frac{1}{\sqrt{\lambda_1 - a}}$$

and

$$\|Z^T L Z\| \leq \|Z^T L Z - Z^T \mathcal{L} Z\| + \|Z^T \mathcal{L} Z\| \overset{(5.32)}{\leq} \|L - \mathcal{L}\| + 2\lambda_1.$$

It follows that

$$\begin{aligned}
&\|\mathcal{Q}^{-1} Z^T \mathcal{L} Z \mathcal{Q}^{-1} - Q^{-1} Z^T L Z Q^{-1}\| \leq \\
&\frac{4\lambda_1 \cdot \|D - \mathcal{D}\|}{(\lambda_1 - a)^2} + \left( \frac{\sqrt{2} \cdot \|D - \mathcal{D}\|}{(\lambda_1 - a)^2} + \frac{1}{\lambda_1 - a} \right) \cdot \|\mathcal{L} - L\| + \\
&\left( \frac{2 \cdot \|D - \mathcal{D}\|^2}{(\lambda_1 - a)^3} + \frac{\sqrt{2} \cdot \|D - \mathcal{D}\|}{(\lambda_1 - a)^2} \right) \cdot (\|L - \mathcal{L}\| + 2\lambda_1) \leq \\
&\frac{8\lambda_1 \cdot \|D - \mathcal{D}\|}{(\lambda_1 - a)^2} + (\frac{4 \cdot \|D - \mathcal{D}\|}{(\lambda_1 - a)^2} + \frac{1}{\lambda_1 - a}) \cdot \|\mathcal{L} - L\| + \\
&\frac{2 \cdot \|D - \mathcal{D}\|^2}{(\lambda_1 - a)^3} \cdot (\|L - \mathcal{L}\| + 2\lambda_1).
\end{aligned} \tag{5.59}$$

If (5.37) and (5.38) hold, then, after combining (5.52), (5.53), (5.54), (5.57), (5.59) and using that $\lambda_1 - a > \lambda_1/2$, which follows from (5.13), we end up with

$$\begin{aligned}
&\min_{U\in\mathbb{R}^{k\times k}: U^TU=UU^T=I_k} \|Z\mathcal{Q}^{-1}\mathcal{X} - ZQ^{-1}XU\|_F \\
&\leq \frac{\mathrm{const}(C,r)\cdot k^2}{n(c-d)\sqrt{\lambda_1-a}}\left(\sqrt{a\cdot n\ln n} + \frac{a\cdot n\ln n}{\lambda_1-a} + \frac{(a\cdot n\ln n)^{3/2}}{(\lambda_1-a)^2}\right) + \\
&\frac{\mathrm{const}(C,r)}{\sqrt{\lambda_1-a}}\cdot\frac{\sqrt{k}\cdot\sqrt{a\cdot n\ln n}}{\lambda_1-a}
\end{aligned}$$

for some $\mathrm{const}(C,r)$. Using that $\sqrt{a\cdot n\ln n} \leq \sqrt{k\cdot a\cdot n\ln n} \leq \frac{\widehat{C}_2}{1+M}(\lambda_1-a) \leq \widehat{C}_2(\lambda_1-a)$ due to (5.13), for some $\widehat{C}_2 = \widehat{C}_2(C,r)$ that we will specify shortly (we will choose it smaller than 1), we can simplify this bound such that

$$\begin{aligned}
&\min_{U\in\mathbb{R}^{k\times k}: U^TU=UU^T=I_k} \|Z\mathcal{Q}^{-1}\mathcal{X} - ZQ^{-1}XU\|_F \\
&\leq \frac{\mathrm{const}(C,r)\cdot k^2}{n(c-d)\sqrt{\lambda_1-a}}\cdot\sqrt{a\cdot n\ln n} + \frac{\mathrm{const}(C,r)}{\sqrt{\lambda_1-a}}\cdot\frac{\sqrt{k}\cdot\sqrt{a\cdot n\ln n}}{\lambda_1-a}.
\end{aligned} \tag{5.60}$$

Similarly to Part 3, we use Lemma 5.3 in [110] to complete the proof of Theorem 5.4.1 for Algorithm 13. Assume that (5.60) holds and let $U \in \mathbb{R}^{k\times k}$ be an orthogonal matrix attaining the minimum, that is we have

$$\begin{aligned}
&\|Z\mathcal{Q}^{-1}\mathcal{X}U^T - ZQ^{-1}X\|_F = \|Z\mathcal{Q}^{-1}\mathcal{X} - ZQ^{-1}XU\|_F \\
&\leq \frac{\mathrm{const}(C,r)\cdot k^2}{n(c-d)\sqrt{\lambda_1-a}}\cdot\sqrt{a\cdot n\ln n} + \frac{\mathrm{const}(C,r)}{\sqrt{\lambda_1-a}}\cdot\frac{\sqrt{k}\cdot\sqrt{a\cdot n\ln n}}{\lambda_1-a}.
\end{aligned} \tag{5.61}$$

As we have noted above, we can choose $\mathcal{X}$ in such a way that $Z\mathcal{Q}^{-1}\mathcal{X} = \frac{1}{\sqrt{\lambda_1-a}}\cdot T$ with $T$ as in Lemma 5.9.4. According to Lemma 5.9.4, if we denote the $i$-th row of $\frac{1}{\sqrt{\lambda_1-a}}\cdot T$ by $\tilde{t}_i$, then $\tilde{t}_i = \tilde{t}_j$ if the vertices $i$ and $j$ are in the same cluster and $\|\tilde{t}_i - \tilde{t}_j\| = \sqrt{\frac{2k}{n(\lambda_1-a)}}$ if the vertices $i$ and $j$ are not in the same cluster. Since multiplying $\frac{1}{\sqrt{\lambda_1-a}}\cdot T$ by $U^T$ from the right side has the effect of applying an orthogonal transformation to the rows of $\frac{1}{\sqrt{\lambda_1-a}}\cdot T$,

the same properties are true for the matrix $\frac{1}{\sqrt{\lambda_1 - a}} \cdot TU^T$. Lemma 5.3 in [110] guarantees that for any $\delta \leq \sqrt{\frac{2k}{n(\lambda_1 - a)}}$, if

$$\frac{16 + 8M}{\delta^2} \left\| \frac{1}{\sqrt{\lambda_1 - a}} \cdot TU^T - ZQ^{-1}X \right\|_F^2 < \underbrace{|C_l|}_{=\frac{n}{k}}, \quad l \in [k], \tag{5.62}$$

with $|C_l|$ being the size of cluster $C_l$, then a $(1 + M)$-approximation algorithm for $k$-means clustering applied to the rows of the matrix $ZQ^{-1}X$ returns a clustering that misclassifies at most

$$\frac{4(4 + 2M)}{\delta^2} \left\| \frac{1}{\sqrt{\lambda_1 - a}} \cdot TU^T - ZQ^{-1}X \right\|_F^2 \tag{5.63}$$

many vertices. If we choose $\delta = \sqrt{\frac{2k}{n(\lambda_1 - a)}}$, then for a small enough $\widehat{C}_2 = \widehat{C}_2(C, r)$ (chosen smaller than 1 and also so small that (5.48) implies (5.49)), because of (5.61), the condition (5.13) implies (5.62). Also, for a large enough $\widetilde{C}_2 = \widetilde{C}_2(C, r)$ the expression (5.63) is upper bounded by the expression (5.14).

## 5.10 Why Running Standard Spectral Clustering on Each Group $V_s$ Separately is not a Good Idea

One might think that the following was a good idea for partitioning $V = V_1 \dot{\cup} \ldots \dot{\cup} V_h$ into $k$ clusters such that every cluster has a high balance value: we could try to run standard spectral clustering with $k$ clusters on each of the groups $V_s$, $s \in [h]$, separately and then to merge the $k \cdot h$ many clusters to end up with $k$ clusters.

The graph shown in Figure 5.8 illustrates that such an approach, in general, fails to recover an underlying fair ground-truth clustering, even when standard spectral clustering succeeds. We have $V = [12]$ and two groups $V_1 = \{1, 2, 3, 7, 8, 9\}$ (shown in red) and $V_2 = \{4, 5, 6, 10, 11, 12\}$ (shown in blue). We want to partition $V$ into two clusters. It can be verified that a clustering with minimum RatioCut value is given by $V =$

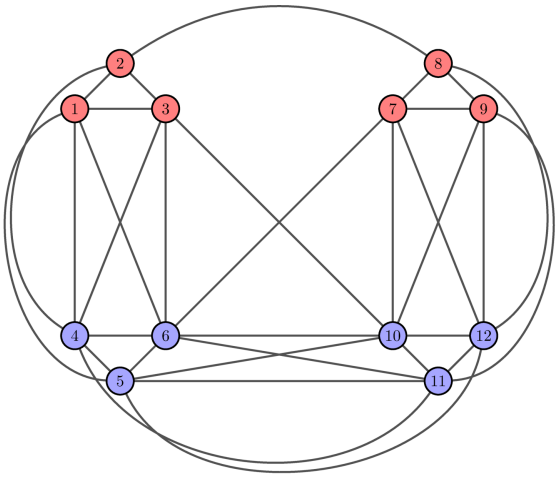



Figure 5.8: Example of a graph for which both standard spectral clustering and our fair versions are able to recover the fair meaningful ground-truth clustering while a naive approach that runs standard spectral clustering on each group separately fails to do so. It is $V = [12]$, $V_1 = \{1, 2, 3, 7, 8, 9\}$, $V_2 = \{4, 5, 6, 10, 11, 12\}$ and the fair ground-truth clustering is $V = \{1, 2, 3, 4, 5, 6\} \dot{\cup} \{7, 8, 9, 10, 11, 12\}$.

$\{1, 2, 3, 4, 5, 6\} \dot{\cup} \{7, 8, 9, 10, 11, 12\}$ and that this clustering is found by running standard spectral clustering. This clustering is perfectly fair with balance$(\{1, 2, 3, 4, 5, 6\}) =$ balance$(\{7, 8, 9, 10, 11, 12\}) = 1$ and is also returned by our fair versions of spectral clustering. Let us now look at the idea of running standard spectral clustering on $V_1$ and $V_2$ separately: when running spectral clustering on the subgraph induced by $V_1$, we obtain the clustering $V_1 = \{1, 2, 3\} \dot{\cup} \{7, 8, 9\}$ as we would hope for. However, in the subgraph induced by $V_2$ the clustering $V_2 = \{4, 5, 6\} \dot{\cup} \{10, 11, 12\}$ does not have minimum RatioCut value and is not returned by spectral clustering. Consequently, no matter how we merge the two clusters for $V_1$ and the two clusters for $V_2$, we do not end up with the clustering $V = \{1, 2, 3, 4, 5, 6\} \dot{\cup} \{7, 8, 9, 10, 11, 12\}$.

Note that for these findings to hold we do not require the specific graph shown in Figure 5.8. The key is its structure: let $V_1 = \{1, 2, 3, 7, 8, 9\}$, $V_2 = \{4, 5, 6, 10, 11, 12\}$, $C_1 = \{1, 2, 3, 4, 5, 6\}$ and $C_2 = \{7, 8, 9, 10, 11, 12\}$. Then the graph looks like a realization of the following random graph model: as in our variant of the stochastic block model introduced in Section 5.4, two vertices $i$ and $j$ are connected with an edge with a certain

probability $\Pr(i,j)$, which is now given by

$$\Pr(i,j) = \begin{cases} a, & i,j \in C_1 \ \vee \ i,j \in C_2 \ \vee \ i,j \in V_2, \\ b, & \text{else}, \end{cases}$$

with $a$ large and $b$ small.

### 5.11 Discussion

In this work, we presented an algorithmic approach towards incorporating fairness constraints into the SC framework. We provided a rigorous analysis of our algorithms and proved that they can recover fair ground-truth clusterings in a natural variant of the stochastic block model. Furthermore, we provided strong empirical evidence that often in real data sets, it is possible to achieve higher demographic proportionality at minimal additional cost in the clustering objective.

An important direction for future work is to understand the price of fairness in the SC framework if one needs to satisfy the fairness constraints exactly. One way to achieve this would be to run the fair $k$-means algorithm of [30] in the last step of our Algorithms 12 or 13. We want to point out that the algorithm of [30] currently does not extend beyond two groups of the same size. Second, our experimental results on the stochastic block model provide evidence that our algorithms are robust to moderate levels of perturbations in the group assignments. Characterizing this robustness rigorously is an intriguing open problem.

## REFERENCES


[1] M. Blum and S. Vempala, "Publishable humanly usable secure password creation schemas," in *AAAI Conference on Human Computation and Crowdsourcing, HCOMP*, 2015, pp. 32–41.

[2] F. Chierichetti, R. Kumar, S. Lattanzi, and S. Vassilvitskii, "Fair clustering through fairlets," in *Advances in Neural Information Processing Systems*, 2017, pp. 5029–5037.

[3] M. Mazurek, S. Komanduri, T. Vidas, L. Bauer, N. Christin, L. Cranor, P. Kelley, R. Shay, and B. Ur, "Measuring password guessability for an entire university," in *ACM SIGSAC Conference on Computer & Communications Security*, ACM, 2013, pp. 173–186.

[4] J Bonneau, "The science of guessing: Analyzing an anonymized corpus of 70 million passwords," in *Security and Privacy (SP)*, IEEE, 2012, pp. 538–552.

[5] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang, "The tangled web of password reuse.," in *NDSS*, vol. 14, 2014, pp. 23–26.

[6] R. Weiss and A. De Luca, "Passshapes: Utilizing stroke based authentication to increase password memorability," in *Nordic Conference on Human-Computer Interaction: Building Bridges*, ACM, 2008, pp. 383–392.

[7] D. Florencio and C. Herley, "A large-scale study of web password habits," in *International Conference on World Wide Web*, ACM, 2007, pp. 657–666.

[8] BBC, *Equifax to be investigated by FCA over data breach*, `http://www.bbc.com/news/technology-41737241`, 2017.

[9] Forbes, *Ebay suffers massive security breach, all users must change their passwords*, `http://www.forbes.com/sites/gordonkelly/2014/05/21/ebay-suffers-massive-security-breach-all-users-must-their-change-passwords/`, 2014.

[10] Adobe, *Important customer security announcement*, `http://blogs.adobe.com/conversations/2013/10/important-customer-security-announcement.html`, 2013.

[11] LinkedIn, *An update on linkedin member passwords compromised*, http://blog.linkedin.com/2012/06/06/linkedin-member-passwords-compromised/, 2012.

[12] Zappos, *Zappos customer accounts breached.* http://www.usatoday.com/tech/news/story/2012-01-16/mark-smith-zappos-breach-tips/52593484/1, 2012.

[13] J. Blocki, S. Komanduri, L. Cranor, and A. Datta, "Spaced repetition and mnemonics enable recall of multiple strong passwords," in *Annual Network and Distributed System Security Symposium, NDSS*, 2015.

[14] J. Blocki, M. Blum, and A. Datta, "Naturally rehearsing passwords," in *Advances in Cryptology - ASIACRYPT - International Conference on the Theory and Application of Cryptology and Information Security*, 2013.

[15] J. Blocki, M. Blum, A. Datta, and S. Vempala, "Towards human computable passwords," *arXiv preprint arXiv:1404.0024*, 2014.

[16] J. Angwin, J. Larson, S. Mattu, and L. Kirchner, *Machine bias - propublica*, https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing, 2018.

[17] M. Kay, C. Matuszek, and S. A. Munson, "Unequal representation and gender stereotypes in image search results for occupations," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ACM, 2015, pp. 3819–3828.

[18] J. Buolamwini and T. Gebru, "Gender shades: Intersectional accuracy disparities in commercial gender classification," in *Conference on Fairness, Accountability and Transparency*, 2018, pp. 77–91.

[19] L. Sweeney, "Discrimination in online ad delivery," *Communications of the ACM*, vol. 56, no. 5, pp. 44–54, 2013.

[20] K. Crawford, *The trouble with bias*, Invited Talk by Kate Crawford at NIPS 2017, Long Beach, CA., 2017.

[21] K. Pearson, "On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[22] I. T. Jolliffe, "Principal component analysis and factor analysis," in *Principal component analysis*, Springer, 1986, pp. 115–128.

[23] H. Hotelling, "Analysis of a complex of statistical variables into principal components.," *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[24] S. Raychaudhuri, J. M. Stuart, and R. B. Altman, "Principal components analysis to summarize microarray experiments: Application to sporulation time series," in *Biocomputing 2000*, World Scientific, 1999, pp. 455–466.

[25] A. F. Iezzoni and M. P. Pritts, "Applications of principal component analysis to horticultural research," *HortScience*, vol. 26, no. 4, pp. 334–338, 1991.

[26] A. Ben-Tal and A. Nemirovski, *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. Siam, 2001, vol. 2.

[27] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[28] F. Chierichetti, R. Kumar, S. Lattanzi, and S. Vassilvitskii, "Fair clustering through fairlets," in *Neural Information Processing Systems (NIPS)*, 2017.

[29] C. Rösner and M. Schmidt, "Privacy preserving clustering with constraints," in *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2018.

[30] M. Schmidt, C. Schwiegelshohn, and C. Sohler, "Fair coresets and streaming algorithms for fair k-means clustering," arXiv:1812.10854 [cs.DS], 2018.

[31] A. Backurs, P. Indyk, K. Onak, B. Schieber, A. Vakilian, and T. Wagner, "Scalable fair clustering," in *International Conference on Machine Learning*, 2019.

[32] N. Stewart, C. Ungemach, A. Harris, D. Bartels, B. Newell, G. Paolacci, and J. Chandler, "The average laboratory samples a population of 7,300 amazon mechanical turk workers," *Judgment and Decision making*, vol. 10, no. 5, p. 479, 2015.

[33] M. Buhrmester, T. Kwang, and S. Gosling, "Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data?" *Perspectives on Psychological Science*, vol. 6, no. 1, pp. 3–5, 2011.

[34] F. Bentley, N. Daskalova, and B. White, "Comparing the reliability of amazon mechanical turk and survey monkey to traditional market research surveys," in *CHI Conference Extended Abstracts on Human Factors in Computing Systems*, ACM, 2017, pp. 1092–1099.

[35] E. Peer, L. Brandimarte, S. Samat, and A. Acquisti, "Beyond the turk: Alternative platforms for crowdsourcing behavioral research," *Journal of Experimental Social Psychology*, vol. 70, pp. 153–163, 2017.

[36] P. Gasti and K. Rasmussen, "On the security of password manager database formats," in *European Symposium on Research in Computer Security*, Springer, 2012, pp. 770–787.

[37] LastPass, *Lastpass security notice*, `https://blog.lastpass.com/2015/06/lastpass-security-notice.html/`, 2015.

[38] OneLOgin, *Security incident*, `https://www.onelogin.com/blog/may-31-2017-security-incident`, 2017.

[39] P. Pimsleur, "A memory schedule," *The Modern Language Journal*, vol. 51, no. 2, pp. 73–75, 1967.

[40] P. A. Wozniak and E. J. Gorzelanczyk, "Optimization of repetition spacing in the practice of learning," *Acta Neurobiologiae Experimentalis*, vol. 54, pp. 59–62, 1994.

[41] S. Suri, D. G. Goldstein, and W. A. Mason, "Honesty in an online labor market.," *Human Computation*, vol. 11, no. 11, 2011.

[42] W. Cheswick, "Rethinking passwords," *Communications of the ACM*, vol. 56, no. 2, pp. 40–44, 2013.

[43] J. Jonides, S. Lacey, and D. Nee, "Processes of working memory in mind and brain," *Current Directions in Psychological Science*, vol. 14, no. 1, pp. 2–5, 2005.

[44] T. Seitz, M. Hartmann, J. Pfab, and S. Souque, "Do differences in password policies prevent password reuse?" In *CHI Conference on Human Factors in Computing Systems, Extended Abstracts.*, 2017, pp. 2056–2063.

[45] S. Furnell, "An assessment of website password practices," *Computers & Security*, vol. 26, no. 7, pp. 445–451, 2007.

[46] R. Shay, S. Komanduri, A. L. Durity, P. S. Huh, M. L. Mazurek, S. M. Segreti, B. Ur, L. Bauer, N. Christin, and L. F. Cranor, "Can long passwords be secure and usable?" In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2014, pp. 2927–2936.

[47] B. Ur, S. Komanduri, R. Shay, S. Matsumoto, L. Bauer, N. Christin, L. F. Cranor, P. G. Kelley, M. L. Mazurek, and T. Vidas, "Poster: The art of password creation," in *Proc. IEEE Symposium on Security and Privacy*, 2013.

[48] B. Grawemeyer and H. Johnson, "Using and managing multiple passwords: A week to a view," *Interacting with Computers*, vol. 23, no. 3, pp. 256–267, 2011.

[49] D. Pilar, A. Jaeger, C. Gomes, and L. Stein, "Passwords usage and human memory limitations: A survey across age and educational background," *PloS one*, vol. 7, no. 12, e51067, 2012.

[50] R. E. Schapire, "The strength of weak learnability," *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[51] S. A. G. M. J. Kearns and R. E. Schapire, "On the sample complexity of weak learning," *COLT Proceedings 1990*, p. 217, 2012.

[52] D. P. Helmbold and M. K. Warmuth, "On weak learning," *Journal of Computer and System Sciences*, vol. 50, no. 3, pp. 551–573, 1995.

[53] R. Shay, S. Komanduri, P. G. Kelley, P. G. Leon, M. L. Mazurek, L. Bauer, N. Christin, and L. F. Cranor, "Encountering stronger password requirements: User attitudes and behaviors," in *Proceedings of the Sixth Symposium on Usable Privacy and Security*, ACM, 2010, p. 2.

[54] C. Cooper, "On the distribution of rank of a random matrix over a finite field," *Random Structures and Algorithms*, vol. 17, no. 3-4, pp. 197–212, 2000.

[55] Twitter, *Jacky lives: Google photos, y'all fucked up. My friend's not a gorilla.* `https : / / twitter . com / jackyalcine / status / 615329515909156865`, 2015.

[56] T. Simonite, *When it comes to gorillas, google photos remains blind*, `https://www.wired.com/story/when-it-comes-to-gorillas-google-photos-remains-blind/`, 2018.

[57] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumeé III, and K. Crawford, "Datasheets for datasets," *arXiv preprint arXiv:1803.09010*, 2018.

[58] R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork, "Learning fair representations," in *International Conference on Machine Learning*, 2013, pp. 325–333.

[59] A. Beutel, J. Chen, Z. Zhao, and E. Huai-hsin Chi, "Data decisions and theoretical implications when adversarially learning fair representations," *CoRR*, vol. abs/1707.00075, 2017.

[60] F. Calmon, D. Wei, B. Vinzamuri, K. N. Ramamurthy, and K. R. Varshney, "Optimized pre-processing for discrimination prevention," in *Advances in Neural Information Processing Systems*, 2017, pp. 3992–4001.

[61] D. Madras, E. Creager, T. Pitassi, and R. Zemel, "Learning adversarially fair and transferable representations," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 3384–3393.

[62] B. H. Zhang, B. Lemoine, and M. Mitchell, "Mitigating unwanted biases with adversarial learning," *arXiv preprint arXiv:1801.07593*, 2018.

[63] M. Olfat and A. Aswani, "Convex formulations for fair principal component analysis," *arXiv preprint arXiv:1802.03765*, 2018.

[64] D. Pedreshi, S. Ruggieri, and F. Turini, "Discrimination-aware data mining," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2008, pp. 560–568.

[65] F. Kamiran, T. Calders, and M. Pechenizkiy, "Discrimination aware decision tree learning," in *Proceedings of the 10th IEEE International Conference on Data Mining*, 2010, pp. 869–874.

[66] T. Calders and S. Verwer, "Three naive Bayes approaches for discrimination-free classification," *Data Mining and Knowledge Discovery*, vol. 21, no. 2, pp. 277–292, 2010.

[67] F. Kamiran and T. Calders, "Data preprocessing techniques for classification without discrimination," *Knowledge and Information Systems*, vol. 33, no. 1, pp. 1–33, 2011.

[68] B. T. Luong, S. Ruggieri, and F. Turini, "K-NN as an implementation of situation testing for discrimination discovery and prevention," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2011, pp. 502–510.

[69] F. Kamiran, A. Karim, and X. Zhang, "Decision theory for discrimination-aware classification," in *Proceedings of the 12th IEEE International Conference on Data Mining*, 2012, pp. 924–929.

[70] T. Kamishima, S. Akaho, H. Asoh, and J. Sakuma, "Fairness-aware classifier with prejudice remover regularizer," in *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, 2012, pp. 35–50.

[71] S. Hajian and J. Domingo-Ferrer, "A methodology for direct and indirect discrimination prevention in data mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 7, pp. 1445–1459, 2013.

[72] M. Feldman, S. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian, "Certifying and removing disparate impact," in *Proceedings of the 21th ACM*

*SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 259–268.

[73] M. Zafar, I. Valera, M. Gomez-Rodriguez, and K. Gummadi, "Fairness constraints: A mechanism for fair classification," *CoRR*, vol. abs/1507.05259, 2015.

[74] B. Fish, J. Kun, and Á. D. Lelkes, "A confidence-based approach for balancing fairness and accuracy," in *Proceedings of the 16th SIAM International Conference on Data Mining*, 2016, pp. 144–152.

[75] P. Adler, C. Falk, S. Friedler, G. Rybeck, C. Scheidegger, B. Smith, and S. Venkatasubramanian, "Auditing black-box models for indirect influence," in *Proceedings of the 16th International Conference on Data Mining*, 2016, pp. 1–10.

[76] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," in *Proceedings of the 3rd innovations in theoretical computer science conference*, ACM, 2012, pp. 214–226.

[77] M. Hardt, E. Price, N. Srebro, *et al.*, "Equality of opportunity in supervised learning," in *Advances in neural information processing systems*, 2016, pp. 3315–3323.

[78] J. Kleinberg, S. Mullainathan, and M. Raghavan, "Inherent trade-offs in the fair determination of risk scores," *arXiv preprint arXiv:1609.05807*, 2016.

[79] A. Chouldechova, "Fair prediction with disparate impact: A study of bias in recidivism prediction instruments," *Big data*, vol. 5, no. 2, pp. 153–163, 2017.

[80] M. Joseph, M. Kearns, J. H. Morgenstern, and A. Roth, "Fairness in learning: Classic and contextual bandits," in *Advances in Neural Information Processing Systems*, 2016, pp. 325–333.

[81] S. Kannan, M. Kearns, J. Morgenstern, M. M. Pai, A. Roth, R. V. Vohra, and Z. S. Wu, "Fairness incentives for myopic agents," in *Proceedings of the 2017 ACM Conference on Economics and Computation*, 2017, pp. 369–386.

[82] D. Ensign, S. A. Friedler, S. Neville, C. E. Scheidegger, and S. Venkatasubramanian, "Runaway feedback loops in predictive policing," *Workshop on Fairness, Accountability, and Transparency in Machine Learning*, 2017.

[83] D. Ensign, S. A. Friedler, S. Neville, C. Scheidegger, and S. Venkatasubramanian, "Runaway feedback loops in predictive policing," *arXiv preprint arXiv:1706.09847*, 2017.

[84] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

[85] Z. C. Lipton, A. Chouldechova, and J. McAuley, "Does mitigating ML's disparate impact require disparate treatment?" *arXiv preprint arXiv:1711.07076*, 2017.

[86] L. C. Lau, R. Ravi, and M. Singh, *Iterative methods in combinatorial optimization*. Cambridge University Press, 2011, vol. 46.

[87] S. Arora, E. Hazan, and S. Kale, "The multiplicative weights update method: A meta-algorithm and applications.," *Theory of Computing*, vol. 8, no. 1, pp. 121–164, 2012.

[88] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," University of Massachusetts, Amherst, Tech. Rep. 07-49, 2007.

[89] I.-C. Yeh and C.-h. Lien, "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2473–2480, 2009.

[90] M. Afifi and A. Abdelhamed, "Afif4: Deep gender classification based on adaboost-based fusion of isolated facial features and foggy faces," *arXiv preprint arXiv:1706.04277*, 2017.

[91] U. Tantipongpipat, S. Samadi, M. Singh, J. H. Morgenstern, and S. Vempala, "Multi-criteria dimensionality reduction with applications to fairness," in *Advances in Neural Information Processing Systems*, 2019, pp. 15 135–15 145.

[92] M. Kaneko and K. Nakamura, "The nash social welfare function," *Econometrica: Journal of the Econometric Society*, pp. 423–435, 1979.

[93] J. F. Nash Jr, "The bargaining problem," *Econometrica: Journal of the Econometric Society*, pp. 155–162, 1950.

[94] M. Hardt, E. Price, N. Srebro, *et al.*, "Equality of opportunity in supervised learning," in *Advances in neural information processing systems*, 2016, pp. 3315–3323.

[95] L. E. Celis, V. Keswani, D. Straszak, A. Deshpande, T. Kathuria, and N. K. Vishnoi, "Fair and diverse dpp-based data summarization," in *International Conference on Machine Learning (ICML)*, 2018.

[96] S. Samadi, U. Tantipongpipat, J. Morgenstern, M. Singh, and S. Vempala, "The price of fair pca: One extra dimension," in *Neural Information Processing Systems (NIPS)*, 2018.

[97] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[98] L. Hagen and A. B. Kahng, "New spectral methods for ratio cut partitioning and clustering," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 11, no. 9, pp. 1074–1085, 1992.

[99] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[100] H. Lütkepohl, *Handbook of Matrices*. Wiley & Sons, 1996.

[101] S. X. Yu and J. Shi, "Segmentation given partial grouping constraints," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 173–183, 2004.

[102] J. Kawale and D. Boley, "Constrained spectral clustering using L1 regularization," in *SIAM International Conference on Data Mining (SDM)*, 2013.

[103] L. Xu, W. Li, and D. Schuurmans, "Fast normalized cut with linear constraints," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[104] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Social Networks*, vol. 5, pp. 109–137, 1983.

[105] S. Ahmadian, A. Norouzi-Fard, O. Svensson, and J. Ward, "Better guarantees for k-means and euclidean k-median by primal-dual algorithms," in *Symposium on Foundations of Computer Science (FOCS)*, 2017.

[106] A. Kumar, Y. Sabharwal, and S. Sen, "A simple linear time $(1+\varepsilon)$-approximation algorithm for $k$-means clustering in any dimensions," in *Symposium on Foundations of Computer Science (FOCS)*, 2004.

[107] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Symposium on Discrete Algorithms (SODA)*, 2007.

[108] P. Sarkar and P. J. Bickel, "Role of normalization in spectral clustering for stochastic blockmodels," *The Annals of Statistics*, vol. 43, no. 3, pp. 962–990, 2015.

[109] C. Davis and W. M. Kahan, "The rotation of eigenvectors by a perturbation. III," *SIAM Journal on Numerical Analysis*, vol. 7, no. 1, pp. 1–46, 1970.

[110] J. Lei and A. Rinaldo, "Consistency of spectral clustering in stochastic block models," *The Annals of Statistics*, vol. 43, no. 1, pp. 215–237, 2015.

[111] M. C. V. Nascimento and A. C. P. L. F. de Carvalho, "Spectral methods for graph clustering - A survey," *European Journal of Operational Research*, vol. 211, no. 2, pp. 221–231, 2011.

[112] S. X. Yu and J. Shi, "Grouping with bias," in *Neural Information Processing Systems (NIPS)*, 2001.

[113] T. Joachims, "Transductive learning via spectral graph partitioning," in *International Conference on Machine Learning (ICML)*, 2003.

[114] Z. Lu and M. A. Carreira-Perpinan, "Constrained spectral clustering through affinity propagation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[115] X. Wang and I. Davidson, "Flexible constrained spectral clustering," in *ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2010.

[116] A. Eriksson, C. Olsson, and F. Kahl, "Normalized cuts revisited: A reformulation for segmentation with linear grouping constraints," *Journal of Mathematical Imaging and Vision*, vol. 39, no. 1, pp. 45–61, 2011.

[117] S. Maji, N. K. Vishnoi, and J. Malik, "Biased normalized cuts," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[118] A. Khoreva, F. Galasso, M. Hein, and B. Schiele, "Learning must-link constraints for video segmentation based on spectral clustering," in *German Conference on Pattern Recognition (GCPR)*, 2014.

[119] X. Wang, B. Qian, and I. Davidson, "On constrained spectral clustering and its applications," *Data Mining and Knowledge Discovery*, vol. 28, no. 1, pp. 1–30, 2014.

[120] M. Cucuringu, I. Koutis, S. Chawla, G. Miller, and R. Peng, "Simple and scalable constrained clustering: A generalized spectral method," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

[121] E. Abbe, "Community detection and stochastic block models: Recent developments," *Journal of Machine Learning Research (JMLR)*, vol. 18, pp. 1–86, 2018.

[122] K. Rohe, S. Chatterjee, and B. Yu, "Spectral clustering and the high-dimensional stochastic blockmodel," *The Annals of Statistics*, vol. 39, no. 4, pp. 1878–1915, 2011.

[123] D. E. Fishkind, D. L. Sussman, M. Tang, J. T. Vogelstein, and C. E. Priebe, "Consistent adjacency-spectral partitioning for the stochastic block model when the model parameters are unknown," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 1, pp. 23–39, 2013.

[124] T. Qin and K. Rohe, "Regularized spectral clustering under the degree-corrected stochastic blockmodel," in *Neural Information Processing Systems (NIPS)*, 2013.

[125] A. Joseph and B. Yu, "Impact of regularization on spectral clustering," *The Annals of Statistics*, vol. 44, no. 4, pp. 1765–1791, 2016.

[126] L. Su, W. Wang, and Y. Zhang, "Strong consistency of spectral clustering for stochastic block models," arXiv:1710.06191 [stat.ME], 2017.

[127] M. Donini, L. Oneto, S. Ben-David, J. Shawe-Taylor, and M. Pontil, "Empirical risk minimization under fairness constraints," in *Neural Information Processing Systems (NeurIPS)*, 2018.

[128] L. E. Celis, L. Huang, and N. K. Vishnoi, "Multiwinner voting with fairness constraints," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[129] L. E. Celis, D. Straszak, and N. K. Vishnoi, "Ranking with fairness constraints," in *International Colloquium on Automata, Languages and Programming (ICALP)*, 2018.

[130] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian, "Certifying and removing disparate impact," in *ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, 2015.

[131] M. B. Zafar, I. Valera, M. G. Rodriguez, and K. P. Gummadi, "Fairness constraints: Mechanisms for fair classification," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[132] R. Mastrandrea, J. Fournet, and A. Barrat, "Contact patterns in a high school: A comparison between data collected using wearable sensors, contact diaries and friendship surveys," *PloS ONE*, vol. 10, no. 9, pp. 1–26, 2015, Data available on `http://www.sociopatterns.org/datasets/high-school-contact-and-friendship-networks/`.

[133] M. R. Weeks, S. Clair, S. P. Borgatti, K. Radda, and J. J. Schensul, "Social networks of drug users in high-risk sites: Finding the connections," *AIDS and Behavior*, vol. 6, no. 2, pp. 193–206, 2002, Data available on `https://sites.google.com/site/ucinetsoftware/datasets/covert-networks/drugnet`.

[134] D. Yan, L. Huang, and M. I. Jordan, "Fast approximate spectral clustering," in *ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2009.

[135] M. Li, X.-C. Lian, J. T.-Y. Kwok, and B.-L. Lu, "Time and space efficient spectral clustering via column sampling," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[136] G. H. Golub and C. F. Van Loan, *Matrix Computations*. John Hopkins University Press, 2013.

[137] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, Eds., *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. Society for Industrial and Applied Mathematics, 2000.

[138] S. Boucheron, G. Lugosi, and O. Bousquet, "Concentration inequalities," in *Advanced Lectures on Machine Learning*, Springer, 2004.

[139] V. Q. Vu and J. Lei, "Minimax sparse principal subspace estimation in high dimensions," *The Annals of Statistics*, vol. 41, no. 6, pp. 2905–2947, 2013.

[140] R. Bhatia, *Matrix Analysis*. Springer, 1997.